132 VM

August 1997

In this issue

update

VM Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: xephon@compuserve.com

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75067
USA
Telephone: 940 455 7050

Australian office
Xephon/RSM
GPO Box 6258
Halifax Street
Adelaide, SA 5000
Australia
Telephone: 088 223 1391

Contributions
Articles published in *VM Update* are paid for at the rate of £170 ($250) per 1000 words for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our *Notes for Contributors*.

Editor
Trevor Eddolls

Disclaimer
Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, EXECs, and other contents of this journal before making any use of it.

Subscriptions and back-issues
A year's subscription to *VM Update*, comprising twelve monthly issues, costs £170.00 in the UK; $255.00 in the USA and Canada; £176.00 in Europe; £182.00 in Australasia and Japan; and £180.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1990 issue, are available separately to subscribers for £14.50 ($21.50) each including postage.

VM Update on-line
Code from *VM Update* can be downloaded from our Web site at http://www.xephon.com; you will need the user-id shown on your address label.

Printed in England.

Sorting large files

GENERAL DESCRIPTION

The sorting time increases rather quickly when the size of the file to be sorted increases. Therefore, to optimize the sorting time for a large file, SORTPART EXEC divides the file into small parts, all of which sort faster than the original file would. Next SORTPART prepares the sorted file parts for merging using FMERGE EXEC – observing FMERGE conventions about merged files.

SORTPART is written in REXX.

SORTPART EXEC USAGE

The user should follow the interactive dialogue to define the sort fields. If there are no sort fields, then all records should be defined as the first and only sort field. Because of CMS restrictions, the sum of the sort fields' length should not be greater than 249 bytes and the input file record length should be less than 32,765 bytes.

The buffer size for SORTPART does not have to be very large because the sort time will slow up. It is quicker to sort a few records and to merge a lot of files than to sort a lot of records and to merge a few files.

SORTPART generates CMS full filenames of sorted parts, suitable to be processed by FMERGE EXEC, which merges up to 256 sorted files at the same time. The source code of FMERGE was published in *VM Update* issue 122, on pages 3-15.

This doesn't mean that you can use only FMERGE EXEC to merge the sorted parts. If you have faster or more powerful merge executable code, you can use that without any problems.

BENCHMARKS

The productivity of SORTPART EXEC is fixed when the real production files, which may vary in size from 16MB to 370MB, are

sorted on a 3380 device with a 3MB per second transfer rate. The number of sorted records is between 62,000 and 1,470,000. In addition, total elapsed sort time is added including merge time, and sort, merge, and total rates are calculated. The merging is achieved with FMERGE EXEC.

The results obtained are shown in Figure 1.

Size (MB)	16	133	370
Records	62,000	1,120,000	1,470,000
Record length (bytes)	264	125	264
Sort length (bytes)	10	10	10
SORTPART buffer (MB)	2	2	2
FMERGE buffer (MB)	2	2	6
Sorted&merged file parts	8	67	185
Sorting (elapsed)	55sec	18min 36sec	32min 25sec
Merging (elapsed)	28sec	5min 42sec	12min 28sec
Total (elapsed)	1min 23sec	24min 18sec	44min 53sec
Sorting rate (MB/min)	17.5	7.1	11.4
Merging rate (MB/min)	34.8	23.3	29.7
Total rate (MB/min)	11.6	5.5	8.2

Figure 1: Results

SORTPART EXEC

```
/*********************************************************************/
/***                                                    ***         ***/
/*** SORTPART      sort parts of file for fmerge         ***  DG'97  ***/
/***                                                    ***         ***/
/*********************************************************************/
```

```
/***   SIZE ØØ146  VER 1.Ø MOD Ø3  TIME 13:16:Ø4  DATE 21/Ø3/97   ***/
/**********************************************************************/

  HI = '1DF8'X
  LO = '1DFØ'X
  CLRSCRN
  DO I = 1 TO 14
    SAY
  END
  SAY'>>>---> Type filename to sort - reply'HI'FN FT FM'LO
  PULL FN FT FM
  SET CMSTYPE HT
  MAKEBUF
  LISTFILE FN FT FM '(' STACK AL
  IF RC ¬= Ø THEN
  DO
    DROPBUF
    SET CMSTYPE RT
    SAY'--- Not found'HI FN FT FM LO
    EXIT
  END
  PULL . . . . BLK BLKS .
  DROPBUF
  SET CMSTYPE RT
  SAY'>>>---> Disk for sorted file parts'
  PULL FM_SORT
  MAKEBUF
  QUERY DISK FM_SORT '(' STACK LIFO
  PULL . . . MODE .
  DROPBUF
  IF MODE ¬= 'R/W' THEN
  DO
    SAY '--- Disk ' FM 'is read/only or not found'
    EXIT
  END
  SAY'>>>--- Type size of buffer in MB'
  PULL BUF_SIZE
  IF DATATYPE(BUF_SIZE) ¬= 'NUM' THEN
  BUF_SIZE = 2
  CLRSCRN
  DO FOREVER
    SAY '--- Sorted fields definition -'
    SAY '    1 - by starting and ending columns'
    SAY '    2 - by starting column and length'
    SAY '    Type 1 or 2'
    PULL ANS
    IF ANS = '1' | ANS = '2' THEN
    LEAVE
  END
  I = Ø
```

```
  DO FOREVER
    IF ANS = '1' THEN
SAY '--- Field - type starting column and ending column or empty line'
    ELSE
    SAY '--- Field - type starting column and length or empty line'
    PULL START_POS END_POS
    IF START_POS = '' THEN
    LEAVE
    IF DATATYPE(START_POS) ¬= 'NUM' | DATATYPE(END_POS) ¬= 'NUM' THEN
    DO
      SAY '--- Input contains not numbers'
      SAY '    Repeat input again'
      ITERATE
    END
    IF ANS = '2' THEN
    END_POS = START_POS + END_POS - 1
    IF START_POS > END_POS | START_POS <= Ø THEN
    DO
      SAY '--- Invalid field definitions'
      SAY '    Repeat input again'
      ITERATE
    END
    FLAG = ''
    DO J = 1 TO I
      IF START_POS <= END.J THEN
        IF START_POS >= BGN.J THEN
      DO
        FLAG = 'E'
        LEAVE
      END
    END
    IF FLAG = 'E' THEN
    DO
      SAY '--- Fields crossing'
      SAY '    Repeat input again'
      ITERATE
    END
    IF END_POS > BLK THEN
    DO
      SAY '--- Field outside input record'
      SAY '    Repeat input again'
      ITERATE
    END
    I = I + 1
    BGN.I = START_POS
    END.I = END_POS
  END
  IF I = Ø THEN
  EXIT
  PART_SIZE = MIN(BLKS, BUF_SIZE * 1Ø24 * 1Ø24 % BLK)
```

```
  N_FILES =  MIN(BLKS, BLKS % PART_SIZE + MIN(1, BLKS // PART_SIZE))
  CLRSCRN
  SAY   '      Unsorted file'HI LEFT(FN,8) LEFT(FT,8) FM LO
  SAY
  SAY   '  Sorted parts from'HI LEFT(FN,8) LEFT('1', 8) FM_SORT LO
  SAY   '                 to'HI LEFT(FN,8) LEFT(N_FILES,8) FM_SORT LO
  SAY
  SAY   '       To be sorted'HI BLKS LO'records in parts size of'    ,
                                      HI PART_SIZE LO'according to'
  SAY   'next sort fields'
  DO J = 1 TO I
    BUF  = '         Field ' J  ' - '   ,
        'Length' HI FORMAT(END.J - BGN.J + 1, 3) LO     ,
        'Start' HI RIGHT(BGN.J,3) LO 'End' HI RIGHT(END.J, 3) LO
    SAY BUF
  END
  SAY ' '
  SAY '--- To start sorting input'HI'1 (YES)'LO
  PULL ANS
  IF ANS ¬= 1 THEN
  EXIT
  SORT_PARM = DMSXMS
  DO J = 1 TO I
    SORT_PARM = SORT_PARM BGN.J END.J
  END
  SAY '    Start sorting ' TIME()
  SAY
  GLOBALV SELECT SORT PUT 'N_FILES'
  DO I = 1 TO N_FILES
 GLOBALV SELECT SORT PUT 'BLK I FM_SORT PART_SIZE FN FT FM SORT_PARM'
    X $ $ A '(' PROF SORTPART W BLK
    GLOBALV SELECT SORT GET 'RC_CODE POSITION'
    IF RC_CODE > Ø THEN
    DO
      SAY '--- Sort terminated due to error'HI RC_CODE POSITION LO
SAY '   The 'I'-th part of' FN FT FM HI'is not sorted'LO'at' TIME()
      SAY
      EXIT
    END
    SAY '   The 'I'-th part of' FN FT FM 'is sorted at' TIME()
  END
```

SORTPART XEDIT

```
/*****************************************************************/
/***                                                  ***         ***/
/*** SORTPART      sort parts of file for fmerge       ***  DG'97  ***/
/***                                                  ***         ***/
/*****************************************************************/
```

```
/***   SIZE 00029  VER 1.0 MOD 01  TIME 13:22:31  DATE 21/03/97   ***/
/*********************************************************************/

  GLOBALV SELECT SORT GET I PART_SIZE SORT_PARM FN FT FM FM_SORT BLK
  RECFM F
  LRECL BLK
  MSGM  OFF
  GET FN FT FM (I-1) * PART_SIZE + 1 PART_SIZE
  RC_CODE = RC
  POSITION = 'when GET'
  ADDRESS CMS GLOBALV SELECT SORT PUT 'RC_CODE POSITION'
  IF RC_CODE ¬= 0 THEN
  QQ
  SORT_PARM
  RC_CODE = RC
  POSITION = 'when SORT'
  ADDRESS CMS GLOBALV SELECT SORT PUT 'RC_CODE POSITION'
  IF RC_CODE ¬= 0 THEN
  QQ
  FF FN I FM_SORT
  RC_CODE = RC
  POSITION = 'when SAVE'
  IF RC_CODE > 1 THEN
  ADDRESS CMS GLOBALV SELECT SORT PUT 'RC_CODE POSITION'
```

SORTPART PREPARATION

The SORTPART EXEC and SORTPART XEDIT must be copied to any accessible mini-disk on a user's virtual machine. Then they will be able quickly to sort files of any size.

Dobrin Goranov
Information Services Co (Bulgaria)

Subscribers who want copies of the code from this issue can call our Web site – http://www.xephon.com – and ask for the article they require. The article will then be e-mailed to them. This service is free to subscribers.

People who want to submit articles for *VM Update* can send them to Trevor Eddolls at any of the addresses on page 2 or e-mail them to xephon@compuserve.com.

CMS back-up/restore – part 3

This month we continue with the code for the CMS back-up/restore system.

```
/*********************************************************************/
/**                  B A C K -U P   F U N C T I O N                 **/
/*********************************************************************/
Backup:
 FullBack = Function_Parm
 Lx   = 13
 If FullBack = 'NO'  Then;
    Call Makeline 6,'Name of include/exclude pair',,8,'-=No file(s)'
 If FullBack = 'YES' Then Do
    Call Makeline 6,'Volume id',,6
    Line.13 = "Center('Ø&Available volumesØ#',79,'-')"
    Do A=1 By 8 Until A >= Words(Mdisk_List)
       LX = LX + 1
       Line.LX = Subword(Mdisk_List,A,8)
       End
    End
 Call Makeline 8,'Sort backup list by',,8,,
                 'User/ VolUser/ VolStart. -=No list'
 Call Makeline 1Ø,'Update compare table',,3,,
                  'Yes/ No.  No=Temporary backup'
 Input.1Ø = 'Yes'
 If Option = 'BATCH' Then Do
    Queue 'RES  6 33' BInput.6
    Queue 'RES  8 33' BInput.8
    Queue 'RES 1Ø 33' BInput.1Ø
    End
Backup_Menu:
 Cursor  = Cur1 '33'
 Call Read
 If Act_Key = 'PF3' Then Signal MainMenu
 If Act_Key = 'ETK' & Accepted = 'YES' Then Signal Run_Backup
 Cur1       = 6
 Accepted   = 'NO'
 Input.8    = Check(Input.8,8,'SORT')
 Input.1Ø   = Check(Input.1Ø,1Ø,'YESNO')
 If Input.8 = ' ' Then Pre.8 = 'Ø('
 If Fullback = 'YES' Then Fname = Check(Input.6,6,'VOLUME')
 If Fullback = 'NO' Then Do
    Line.13  =
    Pre.6    = 'Ø('
    If Input.6 = '-' Then Input.6 = ' '
    Fname    = Input.6
    InRc     = State('INCLUDE')
```

```
    ExRc     = State('EXCLUDE')
    If InRc > Ø & ExRc > Ø Then Do
       Pre.6   = 'Ø%'
       Line.13 =
       Help.6  = 'No INCLUDE and EXCLUDE file found.'
       End
    End
 Call Accept_Check 1Ø
 If Option = 'BATCH' & Accepted = 'NO' Then Do
    Call BatchLog '% ID       ' Help.6
    Call BatchLog '% SORTLIST ' Help.8
    Call BatchLog '% LOGUPDATE' Help.1Ø
    Call Exit Ø
    End
 Signal Backup_Menu
Run_Backup:
 Sort_By      = Translate(Input.8)
 Update_Table = Translate(Input.1Ø)
 Do A=13 To Lx By 1
    Line.A =
    End
 Call Reserve 13 Lx
'MACRO BRDCNTRL PF % RUN Wait'
 If FullBack  = 'YES' Then Do
    If Scan_Directory(Fname) = 'ABEND' Then Signal Abend
    Call Minidisk_Scan 'GETLABEL'
    Selected.Ø = 1
    Selected.1 = '* *' Fname '* *'
    Backtext  = 'Volume' Fname 'selected'
    T_Text    = 'Mount tape for full back-up of' Fname
    InfoText  = Fname '* *' D2C(Cyl_Found) D2C(Word(Mdisk.Fname,3))
    Call Makeline 14,'Cylinders on volume - Total',,
                     'Word(Mdisk.Fname,3)',4
    Call Makeline 15,'Cylinders on volume - Assigned','Cyl_Found',4
    Pre.14 = ' '
    Pre.15 = ' '
    End
 If FullBack = 'NO'  Then Do
    If Scan_Directory('ALL') = 'ABEND' Then Signal Abend
    Call Minidisk_Scan
    If Selected.Ø = Ø Then Signal MainMenu
    Select_Cyls = Ø
    Do A=1 To Selected.Ø By 1
       Select_Cyls = Select_Cyls + Word(Selected.A,6)
       End
    Backtext = Selected.Ø 'of' Dir_found 'mini-disks ('Select_Cyls,
               'of' Cyl_Found 'cylinders) selected'
    T_Text   = 'Mount tape for limited back-up'
    If InRc = Ø Then I_Inname = Fname
                Else I_Inname = '*NONE*'
    If ExRc = Ø Then I_Exname = Fname
```

```
                Else I_Exname = '*NONE*'
     Infotext = '*' D2C(Selected.Ø) D2C(Dir_Found) D2C(Select_Cyls),
                D2C(Cyl_Found) I_Inname I_Exname
     User_Remain = Selected.Ø
     Cyl_Remain  = Select_Cyls
     L1 = Length(Dir_Found)
     L2 = Length(Cyl_Found)
     Call Makeline 14,'Mini-disks scanned','Dir_Found',L1,,
         'Ø#Cylinders:' Cyl_Found
     Call Makeline 15,'Mini-disks selected','Right(Selected.Ø,L1)',L1,,
         'Ø#Cylinders:' Right(Select_Cyls,L2)
     Pre.14 = ' '
     Pre.15 = ' '
     End
  If Option = 'CHECK' Then Do
    'EXEC BRCHKLST'
     Operinfo = "Check file has been created"
     Call Exit Ø
     End
  SaveDate = Packdt()
  Infoline = Savedate Fullback Update_Table InfoText
  Line.12 = "Copies('-',8Ø)"
  Call Reserve 12 15
  Address COMMAND 'EXECIO 1 DISKW CMSBR SELECTED A (FINIS VAR INFOLINE'
  Call TapeCheck T_Text
Dump_Selected:
  Parse Value BRTAPCHK('DUMP') With Rc Vol1 Hdr1
  If Rc = 99 Then Do
     PfLine = '3=Return  4=Exit'
     Enter  = 'IGN'
     Cursor = '21 56'
     Line.21 = ALine('No VOL1 label found - Enter VOL1 labelØ%      Ø^')
     If Option = 'BATCH' Then Queue 'RES 21 56' TapeVol
     Call Read 'NOINPUT'
     Parse Value BRTAPCHK('INIT',Input.21) With Rc Vol1 Hdr1
     IF RC = Ø THEN SIGNAL DUMP_SELECTED
     Push 'Error during TAPE WVOL1'
     SIGNAL ABEND
     End
  If Rc <> Ø Then Do
     Push 'Error during TAPE DUMP'
     Signal Abend
     End
  If Option = 'BATCH' Then Do
     Call Batchlog '% Tape VOL1 label='Vol1
     Call Batchlog '% Back-up running'
     End
 'MACRO BRDCNTRL PF % RUN' Function 'running'
  If Update_Table = 'YES' Then Do
     If Update_Compare_Table_Part1() = 'ABEND' Then Do
        Push 'Insufficient (A)disk space for update of compare table.'
```

```
      Signal Abend
      End
   End
 If Fullback = 'NO' Then Do
   Call Makeline 16,'Mini-disks remaining','Right(User_Remain,L1)',L1
   Call Makeline 17,'Current mdisk owner','User',8
   Call Makeline 18,'Current mdisk cuu','Cuu',4
   Call Makeline 19,'Current mdisk cylinders','Abs(T_Cyl)',5
   Pre.16 = 'Ø&'
   Pre.17 = ' '
   Pre.18 = ' '
   Pre.19 = 'Ø&'
   End
 Do A=1 To Selected.Ø By 1
   Parse Value Selected.A With User Cuu Volid S_Cyl E_Cyl T_Cyl .
   DDR.1 = 'INPUT' Subword(Mdisk.Volid,1,2) Volid
   DDR.2 = 'OUTPUT' Tapecard 'LEAVE COMPACT'
   DDR.4 = 'DUMP FTR Ø TO' (Word(Mdisk.Volid,3) - 1)
   If Fullback = 'NO' Then Do
      User_Remain = User_Remain - 1
      Cyl_Remain  = Cyl_Remain - T_Cyl
      Info.16 = 'Ø#Cylinders:Ø&'Right(Cyl_Remain,L2)
      DDR.4 = 'DUMP FTR' S_Cyl 'TO' E_Cyl
      Call Reserve 16 19
     'MACRO BRDCNTRL REFRESH'
      End
   Call DDR 'BACKUP'
   End
 Address COMMAND 'TAPE RUN'
 Tnum = 1
'FINIS CMSBR LOG A'
'EXECIO * DISKR CMSBR LOG A 1 (FINIS STEM LOG.'
 Do A=1 By 1 While A <= Log.Ø
   If Left(Log.A,1Ø) =  'DUMPING   ' Then Vol = Word(Log.A,2)
   If Space(Log.A,Ø) <> 'STARTSTOPSTARTSTOP' Then Iterate
   Eoj = 'N'
   DO Until Eoj = 'Y'
      Eov = 'N'
      A = A + 1
      If Left(Log.A,1Ø)  = 'END OF JOB' Then Eoj = 'Y'
      If Left(Log.A,13)  = 'END OF VOLUME' Then Do
         A = A + 1
         Eov = 'Y'
         End
      If Datatype(Word(Log.A,1)) <> 'NUM' Then Iterate
      Parse Var Log.A Start Stop .
      Do Start=Abs(Start) To Stop By 1
         T.Vol.Start = Tnum
         End
      If Eov = 'Y' Then Tnum = Tnum + 1
      End
```

```
   End
 If Sort_By <> ' ' Then Call Sort_Select_List Sort_By
 Call Read_Select_List
 Do A=1 To Selected.Ø By 1
   Parse Value Selected.A With . . Vol Scyl .
   Scyl = Abs(Scyl)
   Selected.A = Selected.A T.Vol.SCyl
   End
 If Update_Table = 'YES' Then Do
   If Update_Compare_Table_Part2() = 'ABEND' Then Do
      Push 'Insufficient (A)disk space for update of compare table.'
      Signal Abend
      End
   End
 TotalTape = 'Tape(s) used:' Tnum
 If Option = 'BATCH' Then Call Batchlog '%' TotalTape
 If Sort_By <> ' ' Then Do
   Call Split_Select_List
   Call Print_Select_List
   End
 TotalTape =
'EMSG Back-up successfully completed.'
 Signal MainMenu
/*********************************************************************/
/**              B a c k - u p   s u b r o u t i n e s              **/
/*********************************************************************/
State:
 Procedure EXPOSE Fname Line.13
 If Fname = ' ' Then Return '-1'
 Arg Ft
 Address COMMAND 'STATE' Fname Ft 'A'
 If Rc <> Ø Then Line.13  = "'Ø\Warning:Ø+No" Ft "file found.'"
 Return Rc
/*********************************************************************/
/** "Update_Compare_Table" is split into two parts.                 **/
/** Part1: Calculate the space needed for the new table.            **/
/**        Compare this size with what's left on the disk and abend **/
/**        if there is insufficient.                                **/
/**        (It is stupid to run the back-up and afterwards have to  **/
/**        abend the program because of a disk-write error.)        **/
/**        2Ø2 is the length of a compare table record.             **/
/** Part2: Create a new table.                                      **/
/**        Erase the old one.                                       **/
/**        Rename the new table.                                    **/
/**        This has to wait until after the back-up tapes have been **/
/**        created.                                                 **/
/*********************************************************************/
Update_Compare_Table_Part1:
 Procedure EXPOSE Userinfo.Ø
'QUERY DISK A (STACK'
 Pull . . Check .
```

```
 If Check = 'NOT' Then Return 'ABEND'
 Pull . . . . . Blksize . . Blkleft .
 Blocks = ((Userinfo.Ø * 2Ø2) % Blksize) + 1
 If Blocks >= Blkleft Then Return 'ABEND'
 Return 'OK'
Update_Compare_Table_Part2:
 Do A=1 To Selected.Ø By 1
    Parse Value Selected.A With Userid Cuu Volid,
                           . . . . . Label Volseq .
    NewInfo = Savedate || Left(Volid,6) || Left(Label,6),
                       || Left(Vol1,6)  || D2C(Volseq)
    Parse Value User.Userid.Cuu With . Linenum .
    Parse Value Userinfo.Linenum With Userinfo 28 Log_F 78 Log_L
    IF FULLBACK = 'YES' THEN LOG_F = SUBSTR(LOG_F,26,25)  || NEWINFO
                        ELSE LOG_L = SUBSTR(LOG_L,26,1ØØ) || NEWINFO
    UserInfo.Linenum = Userinfo || Left(Log_F,5Ø) || Log_L
    End
'ERASE CMSBRNEW COMPTBL A'
'EXECIO' Userinfo.Ø 'DISKW CMSBRNEW COMPTBL A 1 F 2Ø2',
        '(FINIS STEM USERINFO.'
 If Rc <> Ø Then Return 'ABEND'
'XEDIT CMSBRNEW COMPTBL A (NOMSG PROFILE BRSORT'
 If Rc <> Ø Then Return 'ABEND'
'ERASE CMSBR COMPTBL A'
 If Rc <> Ø Then Return 'ABEND'
'RENAME CMSBRNEW COMPTBL A CMSBR = = (NOTYPE'
 If Rc <> Ø Then Return 'ABEND'
 Return 'OK'
/*********************************************************************/
/**                 R E S T O R E   F U N C T I O N                 **/
/*********************************************************************/
Restore:
 Restore = Function_Parm
 If Restore = 'DIRECT' Then Do
    Info.4  = 'Ø&Direct'
    Call Makeline 1Ø,'Userid to be restored to',,8
    Call Makeline 12,'Mdisk CUU to be restored to',,4
    Call Makeline 14,'Mdisk WRITE password',,8
    End
 If Restore = 'INDIRECT' Then Do
    Info.4  = 'Ø&Indirect'
    Call Makeline 1Ø,'Userid to be restored to',,8,,
        'If restore is to another user'
    End
 Call Makeline 6,'Userid to be restored from',,8
 Call Makeline 8,'Mdisk CUU to be restore from',,4
 Input.8 = 'Ø191'
 Parse Value BRTDISK('DETACH') With .
 Parse Value BRTDISK('QCUU') With R_Cuu R_Mode
Restore_Menu:
 Cursor = Cur1 '33'
```

```
 Call Read
 Cur1 = 6
 If Act_Key = 'PF3' Then Signal MainMenu
 If Act_Key = 'ETK' & Accepted = 'YES' Then Signal Run_Restore
 Call Restore_Check
 If Restore = 'DIRECT' Then Call Accept_Check 14
                       Else Call Accept_Check 1Ø
 Signal Restore_Menu
/*********************************************************************/
Run_Restore:
 Res_From   = Strip(Input.6)
 Res_Cuu    = Strip(Input.8)
 Res_To     = Strip(Input.1Ø)
 Res_To_Cuu = Strip(Input.12)
 Write_Pass = Strip(Input.14)
 Input.14   = '********'
 Call TapeCheck 'Mount tape for restore,SCAN'
 Call Get_Select_List
 Call Read_Select_List
'DESBUF'
 If Diskr('SELECTED',,Left(Res_From,8) Res_Cuu' ') <> Ø Then Do
   'EMSG Userid and/or mdisk-cuu not on this tape.'
    Signal Restore_Menu
    End
 Pull .
 Pull . . Volid CylStart CylEnd CylTotal R_Type Format .
 If Restore = 'INDIRECT' Then Do
    If Format <> 'CMS' Then Do
      'EMSG Selected mdisk is NOT a CMS formated disk',
           '- Use direct restore.'
       Signal Restore_Menu
       End
    End
 Volseq = '?'
 If Diskr('COMPTBL',,Left(Res_From,8) Res_Cuu' ') = Ø Then Do
    Pull .
    Parse Pull 28 Compinfo
    Do 7
       If Left(Compinfo,6) = TDate Then;
          Volseq = C2D(Substr(Compinfo,25,1))
       Compinfo = Substr(Compinfo,26)
       End
    End
 If Volseq <> 1 Then Do
    If Volseq = '?' Then 'EMSG Cannot determine volseq for this user'
                    Else 'TAPE RUN'
    Call TapeCheck 'Mount volseq' Volseq
    End
'MACRO BRDCNTRL PF % RUN' Function 'running'
 If Restore = 'INDIRECT' Then Do
    Parse Value BRTDISK('DEFINE' R_Type Cyltotal) With Rc Text
```

```
    If Rc <> Ø Then Do
       Push Text
       Signal Abend
       End
    End
 Cylstart = Abs(Cylstart)
 Cylend   = Abs(Cylend)
 If Restore = 'DIRECT' Then Do
    Call Write_Link Write_Pass
    If LinkRC <> Ø Then Do
       Push Text
       Signal Abend
       End
    End
 DDR.1 = 'INPUT' TapeCard 'REWIND'
 DDR.2 = 'OUTPUT' R_Cuu R_Type 'SCRATCH'
 DDR.4 = 'RESTORE' CylStart 'TO' CylEnd 'REORDER Ø'
 Queue 'YES'                              /* In case of MSG DMK725R */
 If Fullback = 'NO' Then Call DDR 'RESTLIM' Volid Right(Cylstart,4,Ø)
                    Else Call DDR 'RESTFULL'
'DESBUF'
 If Restore = 'DIRECT' Then Do
    Parse Value Cp('DETACH' R_Cuu) With . .
   'EMSG Direct restore successfully completed.'
    Signal MainMenu
    End
 Parse Value BRTDISK('ACCESS') With Rc Disk_Files
 If Rc <> Ø Then Do
    Push 'Error during access of temporary mdisk.'
    Signal Abend
    End
 If Disk_Files = Ø Then Do
    Push 'Restored mini-disk does not contain any files.'
    Signal Abend
    End
 If Res_To = ' ' Then Res_To = Res_From
 Botline   = Lscreen - 2
 Scroll_Size = Lscreen - 9
 Selected  = Ø
 Line.     =
 Filename  = '*'
 Filetype  = '*'
 Rinfo.    =
 RPre.     = 'Ø(_Ø#'
 Sort_By   = 'NAME'
New_Files:
 Count  = Ø
 RLine. =
 Line.4 = "'  Ø('Left(Filename,8)'Ø('Left(Filetype,8)'",
        || "Ø+<-- Search criteria, LISTFILE format. Sorted by' Sort_By"
 Line.5 = "'  Ø:FilenameØ:FiletypeØ#  Ø:FormatØ:LreclØ#  ",
```

```
        || "Ø:RecordsØ#   Ø:BlocksØ:  Date     Time  Ø#'"
'EXEC BRLFILE' R_Cuu R_Mode Filename Filetype Sort_By
 If Rc <> Ø Then Signal Abend
 If Queued() > Ø Then Do A=1 To Queued() By 1
    Parse Pull . . Fn Ft . Rinfo
    If RInfo.Fn.Ft <> ' ' Then Rinfo = Rinfo.Fn.Ft
    RLine.A = RPre.Fn.Ft || Left(Fn,8) Left(Ft,8) Right(Rinfo,55)
    End
Restore_Scroll:
 Cur1 =
 Pre_Count = Count
 Do A=6 To (Lscreen-4) By 1
    Count = Count + 1
    Line.A = '"'RLine.Count'"'
    If Cur1 = ' ' Then Do
       If Substr(Line.A,4,1) = '_' Then Cur1 = A
       End
    End
 End_Count = Count
 Line.BotLine = "'Mark (X) the files you wish to restore and",
                "press ENTER.Ø&Restoring to' Res_to "
 EndCheck = End_Count + 1
 If Pre_Count      = Ø      Then Pf7  = '     '
                            Else Pf7  = '7=Bwd'
 If RLine.EndCheck = ' '    Then Pf8  = '     '
                            Else Pf8  = '8=Fwd'
 If Sort_By        = 'DATE' Then Pf12 = '12=Sort/Name'
                            Else Pf12 = '12=Sort/Date'
 PfLine = '3=Return  4=Exit  'Pf7'  'Pf8'  'Pf12
 PfLine = Left(PfLine,58) Right('Ø&Restored:' Selected,17)
 Cur2 = 2
 If Cur1 = ' ' Then Do
    Cur1 = 4
    Cur2 = 4
    End
 If RLine.1 = ' ' Then 'EMSG No match on specified search criteria.'
 Enter  = 'IGN'
 Cursor = Cur1 Cur2
 Call Read 'NOPULL'
 Count = Pre_Count
 Parse Pull Act_Key Num Pos Input
 If Queued() = Ø & Left(Act_Key,2) = 'PF' Then Do
    Operinfo = 'Restore function completed.' Selected 'files restored.'
    If Act_Key = 'PF3'  Then Signal MainMenu
    If Act_Key = 'PF4'  Then Call Exit Ø
    If Act_Key = 'PF7'  Then Count   = Pre_Count - Scroll_Size
    If Act_Key = 'PF8'  Then Count   = End_Count
    If Act_Key = 'PF12' Then Do
       If Sort_By = 'NAME' Then Sort_By = 'DATE'
                           Else Sort_By = 'NAME'
       Signal New_Files
```

```
      End
   Signal Restore_Scroll
   End
If Act_Key = 'RES' Then Push Act_Key Num Pos Input
Old_Filename = Filename
Old_Filetype = Filetype
Restore_List =
Do Queued()
   Pull Key Num Pos Input
   If Key <> 'RES' Then Iterate
   Select
      When Num = 4 Then Do
           If Pos = 4  Then Filename = Input
           If Pos = 13 Then Filetype = Input
           End
      When Input = 'X' Then Restore_List = Restore_List Num
      Otherwise Nop
      End
   End
If Restore_List = ' ' Then Do
   If Filename <> Old_Filename Then Signal New_Files
   If Filetype <> Old_Filetype Then Signal New_Files
   Signal Restore_Scroll
   End
Filename = Old_Filename
Filetype = Old_Filetype
If Left(Act_Key,2) = 'PF' Then Do
  'EMSG File(s) selected and' Act_key 'pressed - Nothing performed.'
   Signal Restore_Scroll
   End
'CP SPOOL PUN TO' Res_To 'NOHOLD NOCONT CL A'
Do A=1 To Words(Restore_List) By 1
   B = (Pre_Count + Word(Restore_List,A)) - 5
   Parse Value RLine.B With . 6 fn ft .
   Address COMMAND 'DISK DUMP' Fn Ft R_Mode
   If Rc <> Ø Then Do
     'BRADPSW' Right(R_Cuu,8,Ø)
     'ACCESS' R_Cuu R_Mode
      Address COMMAND 'DISK DUMP' Fn Ft R_Mode
      If Rc <> Ø Then Do
         Operinfo = 'Restore function terminated. ',
                    || Selected 'files restored.'
        'CP SPOOL PUN OFF'
         Push 'Error during DISK DUMP to' Res_to
         Signal Abend
         End
      End
   Selected = Selected + 1
   Rinfo.Fn.Ft = 'has been sent to' Left(Res_To,35)
   RPre.Fn.Ft  = 'Ø(_Ø^'
   RLine.B = RPre.Fn.Ft || Left(Fn,8) Left(Ft,8) Right(Rinfo.Fn.Ft,55)
```

```
    End
'CP SPOOL PUN OFF'
 Signal Restore_Scroll
/*********************************************************************/
/**              R e s t o r e   s u b r o u t i n e                **/
/*********************************************************************/
Restore_Check:
 If Input.6 = ' ' Then Pre.6 = '0%'
                  Else Pre.6 = '0('
 Input.8 = Check(Input.8,8,'CUU')
 If Input.10 = '=' Then Input.10 = ' '
 If Input.10 = ' ' Then Input.10 = Check(Input.6,10,'USERID')
                   Else Input.10 = Check(Input.10,10,'USERID')
 If Restore = 'INDIRECT' Then Return
 If Input.12 = ' ' | Input.12 = '=' Then Input.12 = Input.8
 Input.12 = Check(Input.12,12,'CUU')
 Pre.14 = '0%'
 If Pre.10   = '0%' Then Return
 If Pre.12   = '0%' Then Return
 If Input.14 = ' '  Then Do
    Help.14  = 'Must be specified'
    Return
    End
 Call Write_Link Input.14
 Parse Value Cp('DETACH' R_Cuu) With .
 If LinkRc = 0 Then Do
    Pre.14 = '0('
    Return
    End
 Help.14 = Strip(Text)
 Return
Write_Link:
 Arg RW_Pw
 Parse Value Cp('LINK' Input.10 Input.12 R_Cuu 'W' Rw_Pw),
       With LinkRc Text
 Return
/*********************************************************************/
/**               L I S T T A P E   F U N C T I O N                 **/
/*********************************************************************/
Listtape:
 Call Makeline 6,'Output device',,8,'Terminal/ Printer'
 Call Makeline 8,'Sort output by',,8,'User/ VolUser/ VolStart'
Listtape_Menu:
 If Pre.8 = '0%' Then Cur1 = 8
 If Pre.6 = '0%' Then Cur1 = 6
 Cursor = Cur1 '33'
 Call Read
 If Act_Key = 'PF3' Then Signal Mainmenu
 If Act_Key = 'ETK' & Accepted = 'YES' Then Signal Run_Listtape
 Input.6 = Check(Input.6,6,'OUTPUT')
 Input.8 = Check(Input.8,8,'SORT')
```

```
 Call Accept_Check 8
 Cur1 = 6
 Signal Listtape_Menu
Run_Listtape:
 Out_Device = Translate(Input.6)
 Sort_By    = Translate(Input.8)
 Call TapeCheck 'Mount tape to be listed,SCAN'
 Call Get_Select_List
Re_Sort:
 Call Sort_Select_List Sort_By
 Call Read_Select_List
 Call Split_Select_List
 If Out_Device = 'PRINTER' Then Do
    Call Print_Select_List
    Signal Mainmenu
    End
 Old_Sort = Sort_By
 Line. =
 Line.3 = "'Ø+'Left(IText1,4Ø) 'Tape created:' Tapedate"
 Line.5 = "'    Ø: Userid Ø/  Ø: CuuØ/  Ø: VolidØ/  Ø:Start",
          || "Ø/ Ø:  EndØ/ Ø:TotalØ/ Ø: TypeØ/ Ø: LabelØ#'"
 A = Lscreen - 3
 Line.A = Copies(' ',65)'Sort:' Sort_By
 Scroll_Size = Lscreen - 8
 Count  = Ø
 Enter  = 'IGN'
 Cursor = '1 1'
Listtape_Scroll:
 Pre_Count = Count
 Do A=6 To (Lscreen-3) By 1
    Count = Count + 1
    Line.A = '"    'Selected.Count'"'
    End
 End_Count = Count
 EndCheck  = End_Count + 1
 If Pre_Count = Ø Then Pf7 = '     '
                  Else Pf7 = '7=Bwd'
 If Selected.EndCheck = ' ' Then Pf8 = '     '
                            Else Pf8 = '8=Fwd'
 PfLine = '3=Return  4=Exit  'Pf7'  'Pf8'  9=Print ',
          '1Ø=Sort/U  11=Sort/VU  12=Sort/VS'
Listtape_Noscroll:
 Call Read
'DESBUF'
 Count = Pre_Count
 Select
    When Act_Key = 'PF3'  Then Signal Mainmenu
    When Act_Key = 'PF7'  Then Count = Pre_Count - Scroll_Size
    When Act_Key = 'PF8'  Then Count = End_Count
    When Act_Key = 'PF9'  Then Do
         Call Print_Select_List
```

```
         Count = Pre_Count
         End
    When Act_Key = 'PF1Ø' Then Sort_By = 'USER'
    When Act_Key = 'PF11' Then Sort_By = 'VOLUSER'
    When Act_Key = 'PF12' Then Sort_By = 'VOLSTART'
    Otherwise Signal Listtape_Noscroll
    End
 If Old_Sort <> Sort_By Then Signal Re_Sort
 Signal Listtape_Scroll
/*********************************************************************/
/**                          L I S T U S E R                        **/
/*********************************************************************/
ListUser:
 Call Makeline 6,'Userid',,8
 Call Makeline 8,'Mdisk CUU',,4,'? will display available CUUs'
 Input.8 = 'Ø191'
 Enter   = 'IGN'
ListUser_Menu:
 If Pre.8 = 'Ø%' Then Cur1 = 8
 If Pre.6 = 'Ø%' Then Cur1 = 6
 Cursor = Cur1 '33'
 Call Read
 Cur1 = 6
 If Act_Key = 'PF3' Then Signal MainMenu
 If Act_Key = 'ETK' & Accepted = 'YES' Then Signal Run_Listtape
 Do A=11 to 22 By 1
    Line.A =
    End
 A = 12
 Input.6 = Check(Input.6,6,'USERID')
 If Input.6 <> ' ' Then Do
    Pre.6 = 'Ø('
    If Input.8 = '?' Then Do
       Pre.8   = 'Ø('
       If Listuser_AllCuu() = Ø Then Do
         'EMSG No information available for userid.'
          Signal Listuser_menu
          End
       Line.11 = "Center('Ø&Available CUUsØ#',79,'-')"
       Cur1 = 8
       Signal Listuser_menu
       End
    End
 If Input.8 <> '?' Then Input.8 = Check(Input.8,8,'CUU')
 If Pre.6 = 'Ø%' Then Signal Listuser_Menu
 If Pre.8 = 'Ø%' Then Signal Listuser_Menu
 User_Cuu = Left(Input.6,8) Input.8
 If Diskr('COMPTBL',,User_Cuu) <> Ø Then Do
   'EMSG No information available for userid/cuu.'
    Signal ListUser_Menu
    End
```

```
 Pull .
 Parse Pull . 28 Info
 Space  = '         '
 Line.11 = "'"Space"Ø:   Date    Ø/Ø:Time Ø/Ø:VolumeØ/Ø:Label Ø/",
           || "Ø:Tape   SeqØ/Ø:Backup Ø/'"
 Numeric Digits 13
 Num. = '999999999999'
 Do A=1 To 7 By 1
    If Left(Info,25) <> ' ' Then Do
       Parse Value Unpkdt(Substr(Info,1,6)) With Date';'Time YMD TTMM
       Parse Value Info With . 7 Volid 13 Label 19 Tlbl 25 Volseq 26
       Num.A = YMD || TTMM
       If A > 2 Then Btype = ' Limited'
                Else Btype = ' Full'
       Text.A = '"'Space'Ø+'Date'  'Time'  'Volid'  'Label'  'Tlbl,
                || Right(C2D(Volseq),4) Btype'"'
       End
    Info = Substr(Info,26)
    End
 Lnum = 12
 Do 7
    Check = Min(Num.1,Num.2,Num.3,Num.4,Num.5,Num.6,Num.7)
    If Check = '999999999999' Then Leave
    Do A=1 To 7 By 1
       If Num.A <> Check Then Iterate
       Lnum = Lnum + 1
       Line.Lnum = Text.A
       Num.A = '999999999999'
       End
    End
 Signal ListUser_Menu
/*********************************************************************/
/**              L i s t u s e r   s u b r o u t i n e              **/
/*********************************************************************/
Listuser_AllCuu:
'FINIS CMSBR COMPTBL A'
 B = Ø
 Goback_Rc = Ø
 Do Forever
    If Diskr('COMPTBL',,Left(Input.6,8),'*','NOFINIS') <> Ø Then Do
       Line.A = '"'Line.A'"'
       Return Goback_Rc
       End
    Goback_Rc = 1
    Pull
    Pull . 1Ø Cuu 14
    B = B + 1
    If B = 14 Then Do
       Line.A = '"'Line.A'"'
       A = A + 1
       B = Ø
```

```
       End
    Line.A = Line.A || Cuu'  '
    End
/*********************************************************************/
/**             G e n e r a l   s u b r o u t i n e s               **/
/*********************************************************************/
Read:
 Arg Read_Opt
 Call_Line = Sigl
 If Read_Opt = 'NOINPUT' Then Brd_Opt = '% NOINPUT'
                         Else Brd_Opt =
 If Enter = 'OK' Then Keys = 'Ø'
                 Else Keys =
 If PfLine <> ' ' Then Do
    SavePf = '* 'PfLine
    Do While SavePf <> ' '
       Parse Value SavePf With . Num'='SavePf
       Keys = Keys Num
       End
    End
 Call Reserve 'ALL'
 If Option = 'BATCH' Then Do
   'MACRO BRDCNTRL REFRESH'
    If Call_Line <> Last_Line Then Sleep = -1
    Else Do
       Sleep = Sleep + 1
       If Sleep > 3Ø Then Sleep = 1Ø
      'CP SLEEP' Sleep 'SEC'
       End
    Last_Line = Call_Line
    PfLine =
    Queue 'ETK'
    End
ReadLoop:
'MACRO BRDCNTRL PF' Pfline '% READ KEYS='Keys 'CURSOR='Cursor Brd_Opt
 If Read_Opt = 'NOPULL' Then Return
 Parse Pull Key Num Pos Input
 If Key = 'PF1' Then Do
   'DESBUF'
   'EXTRACT /CURSOR/'
    A = Cursor.1
    Cursor = Cursor.1 33
    Msg = Help.A
    If Help.A = ' '  Then Msg = 'No help available'
    If Pre.A  = 'Ø%' Then 'EMSG' Msg
    Signal ReadLoop
    End
 If Key             = 'PF4' Then Call Exit Ø
 Act_Key            = '***'
 If Substr(Key,1,2) = 'PF'  Then Act_Key = Key
 If Key             = 'ETK' Then Act_Key = Key
```

```
 If Key             = 'RES' Then Push Key Num Pos Input
 Do Queued()
    Parse Upper Pull Key Num Pos Input
    If Key = 'RES' Then Input.Num = Strip(Input)
    End
 Return
/*********************************************************************/
/**                                                                 **/
/*********************************************************************/
Reserve:
 Parse Arg X Y
 If X = 'ALL' Then Do
    X = 3
    Y = Lscreen - 2
    End
 Do X=X To Y By 1
    Interpret 'SET RESERVED' X 'BLU NON N' Line.X
    End
 If Option = ' '     Then Return
 If Option = 'CHECK' Then 'SET RESERVED -3 N' Right('Ø&CHECK MODE',8Ø)
 If Option = 'BATCH' Then 'SET RESERVED -3 N' Right('Ø&BATCH MODE',8Ø)
 Return
/*********************************************************************
 **                                                                 **
 *********************************************************************/
TapeCheck:
 Parse Arg Usertext','Scantape
 FCuu = 'FREE'
TapeC_Loop:
 Parse Value Cp('QUERY VIRTUAL 181') With Q_Rc . . . . Tape_Cuu .
 If Q_Rc <> Ø Then Do
    If FCuu <> 'FREE' Then Do
       Parse Value Cp('ATTACH' FCuu '* 181') With Q_Rc CpText
       If Q_Rc <> Ø Then 'EMSG Unable to attach' FCuu '-' CpText
       End
    Tape_Cuu = FCuu
    End
 If Q_Rc <> Ø Then Do
    If Option = 'BATCH' Then Do
       Call Batchlog Ø 'No tapedrive attached as 181'
       Call Exit Q_Rc
       End
    FCuu = Word(Cp('QUERY TAPES FREE'),3)
    If FCuu = 'FREE',
       Then TapeText = 'Attach and press ENTER'
       Else TapeText = 'Press ENTER to attachØ&'FCuu'Ø^or attach other'
    TapeText = 'No tapedrive attached as 181 -' TapeText
    End
 If Q_Rc = Ø Then Do
    Tapetext = Usertext '- Press ENTER whenØ&'Tape_Cuu'Ø^is ready'
    Parse Value BRTAPTYP(181) With Rc Devtype Density
```

```
    If Rc <> Ø Then Do
       If Rc = 1 Then Push 'Device 181 is not a tapedrive'
       If Rc = 2 Then Push 'Unknown tapedrive'
       Signal Abend
       End
    TapeCard = '181' Devtype '(MODE' Density
    Address COMMAND 'TAPE MODESET (DEN' Density
    End
 Line.21 = ALine(TapeText)
 PfLine  = '3=Return  4=Exit'
 Enter   = 'OK'
 Cursor  = '2 1'
Tc_Read:
 Operinfo = Function 'terminated due to operator request.'
 Call Read 'NOINPUT'
 If Act_Key = 'PF3'  Then Signal MainMenu
 Operinfo =
 E_Text =
 If Q_Rc <> Ø Then Signal TapeC_Loop
 If Word(Cp('REW 181'),1) <> Ø Then E_Text = 'Tape not ready'
 Else If Scantape = 'SCAN' Then Do
    If Word(BRTAPCHK('VERIFY'),1) <> Ø Then E_Text =,
      'Mounted tape is not volume 1 of a CMS Back-up/Restore tape.'
    End
 If E_Text = ' ' Then Return
'EMSG' E_text
 Signal Tc_Read
/*********************************************************************/
/**                                                                 **/
/*********************************************************************/
Scan_Directory:
 Procedure EXPOSE Fullback Fname Cyl_Found Dir_Found User. UserInfo. Rc
 Arg Scan_For
 User.    = '000000000000'
 Loginfo. =
 New_Date_Time = Substr(Date('S'),3),
              || Space(Translate(Time(),' ',':'),Ø)
 Rc = Diskr('COMPTBL','USERINFO.')
 Do A=1 To Userinfo.Ø By 1
    Parse Value Userinfo.A With U_Id U_Cuu U_Date_Time LogInfo.A
    User.U_Id.U_cuu = U_Date_Time A
    End
 Address COMMAND 'ERASE CMSBR SCANLIST A'
 Parse Value BRDSIEC(Fullback,Fname,Scan_for) With Rc Cyl_Found
 If Rc <> Ø Then Return 'ABEND'
 If Queued() = Ø Then Do
    Push 'No mini-disks selected - All have been excluded.'
    Return 'ABEND'
    End
```

```
 User_Count = Ø
 Do Queued()
    Parse Pull Dasd_Cuu Force_Parm Userline 1 . . Userid Cuu .
    Parse Value User.Userid.Cuu With Old_Date_Time X
    If X <> ' ' Then Do
       LogInfo = Loginfo.X
       If Force_Parm = 'FORCE' Then Old_Date_Time = '000000000000'
       End
    Else Do
       Userinfo.Ø = Userinfo.Ø + 1
       X = Userinfo.Ø
       LogInfo =
       USER.USERID.CUU = '*' X
       End
    Userinfo.X = Left(Userid,8) Cuu New_Date_Time LogInfo
    User_Count = User_Count + 1
    Userline.User_Count = Userline Old_Date_Time || Dasd_Cuu
    End
'EXECIO' User_Count 'DISKW CMSBR SCANLIST A Ø F 57',
       '(FINIS STEM USERLINE.'
 Push 'DISK'
'XEDIT CMSBR SCANLIST A (NOMSG PROFILE BRSORT'
 If Rc > 99000 Then Return 'ABEND'
 Dir_Found = Rc
 Return 'OK'
/*********************************************************************/
/**                                                                 **/
/*********************************************************************/
Minidisk_Scan:
 Parse Arg Brmd_input
'MACRO BRDCNTRL RUN Mini-disk scan running'
 Address COMMAND 'BRMDSCAN' Brmd_Input
 If Rc <> Ø Then Signal Abend
 If Diskr('SELECTED','SELECTED.') = Ø Then Return
 Selected.Ø = Ø
'EMSG No mini-disks selected - No changes since last back-up.'
 Return
/*********************************************************************/
/**                                                                 **/
/*********************************************************************/
```

Editor's note: this article will be continued next month.

Michael Plannthin (Denmark)

Receive all RDR class entries into one CMS file

OVERVIEW

With the RECEIVE command, you can read one entry from your reader into one CMS file. Sometimes, however, it would be preferable to receive together all entries from a reader class into one CMS file. This is what RECEIVAL does.

SYNTAX

RECEIVAL is called with the following parameters:

```
RECEIVAL fn <ft <fm>> <(<APPEND> <CL n>
```

where fn, ft, fm specifies the CMS file to be created; the default for ft is RDR, and for fm filemode A is assumed. If you want to append the reader files to an existing CMS file, you can specify option APPEND. Class ‘n’ is the reader class to work on, the default for the class is A. ‘?’ gives a help panel for the function.

INSTALLATION-SPECIFIC CONFIGURATION

The hardcoded value is RECEIVAL, which internally uses the RECEIVE command and saves the entries into the temporary files R$$CVz R$$CV A, where z runs from 0 to the number of entries in the reader class.

RECEIVAL EXEC

```
/*********************************************************************/
/* Receive all entries of class 'n' into one CMS file                */
/*********************************************************************/
/* Call:   RECEIVAL fn <ft <fm>> <(<APPEND> <CL n>                   */
/*                  fn ft fm          : CMS file to be created       */
/*                                    :   defaults: ft = RDR         */
/*                                    :             fm = A           */
/*                  APPEND            : append to existing CMS file  */
/*                  n                 : reader class to be examined  */
/*                                    :   default: A                 */
```

```
/*********************************************************************/
trace off
parse upper arg fn ft fm '(' parm
if fn = '?' then signal hilfe
'MSGMODE OFF'
parse upper var parm p1 'CL' class p2
if fn = '' then signal help
if ft = '' then ft = 'RDR'
if fm = '' then fm = 'A'
if class = '' then class = 'A'
append = ''
text = 'without'
if p1 = 'APPEND' | p2 = 'APPEND' then do
   append = 'APPEND'
   text = 'with'
end
say '... Receive File' fn ft fm text 'APPEND from RDR-Files Class' class
'SET CMSTYPE HT'
'EXECIO * CP (STR Q V ØØC'
parse pull . . . savclass .
'CP SPOOL RDR CL' class
'ERASE * R$$CV A'
if append ¬= 'APPEND' then 'ERASE' fn ft fm
rc = Ø
z = Ø
do forever
   'RECEIVE = R$$CV' || z 'R$$CV A'
   if rc ¬= Ø then leave
   'COPYFILE R$$CV' || z 'R$$CV A' fn ft fm '(APPEND'
   'COPYFILE' fn ft fm '= = = (OLDD RECFM F LRECL 8Ø'
   'ERASE R$$CV' || z 'R$$CV A'
   z = z + 1
end
'CP SPOOL RDR CL' savclass
'SET CMSTYPE RT'
say '...' z 'RDR-Files received'
exit
/*********************************************************************/
/* Help                                                              */
/*********************************************************************/
hilfe:
'VMFCLEAR'
address cms 'type receival exec * 1 11'
```

Dr Reinhard Meyer (Germany)

Mini-disk cacheing in VM/ESA

One of the best things that came with VM/ESA is probably mini-disk cacheing, or MDC as it is called in VM/ESA manuals. This feature allows you to save I/O to DASD by 'cacheing' disks, or parts of them, in storage.

It first worked on expanded storage only, and from VM/ESA 1.2.2 you were able to use chunks of main storage for cacheing.

Activation of MDC is very simple – once you have decided how much storage you wish to use for cacheing, and what parts of your data you need to cache – all you have to do is issue a couple of CP commands and your MDC is on!

It is so easy that most of the time you allocate too much storage to MDC, and run into a paging problem because the storage chunks come from your DPA! When you are not paging, you are never sure whether you have given MDC all the storage you could.

Guidelines for MDC settings can be found in VM/ESA literature, in the SC24-5521 *VM/ESA Planning and Administration* manual, chapter 21 *Expanded Storage Planning and Administration*.

Once you succeeded in balancing (more or less) your paging/MDC storage utilization, you still can't say that your MDC system is performing at its best. There is no CP command to monitor mini-disk cacheing activity. Even a real-time performance monitor cannot give useful information about MDC (I use Explore for VM from CA, I don't know if another product can).

The information I needed most and couldn't find is how each DASD and each user is performing with MDC. This information exists in CP control blocks, but I know of no CP command to get at it properly.

I had to dig in Real Device Blocks and VMDBKs to get the amounts of SIOs saved by MDC. I did it with two small REXX programs:

- QMDCUSR – to display VM users' MDC performance.
- QMDCDSK – to display DASD MDC performance.

These programs present fullscreen displays, with the ability to sort the information by various criteria and even get ‘hot shots’ sampling in a given time interval.

Using these EXECs, you can easily find which users fit best in your MDC, and which are just wasting your storage.

Three more points to remember:

- MDC is using CPU cycles. Do not cache devices that do not perform well.
- Do not mix cacheing with MDC and with CACHE DASD controllers on the same device.
- These REXX programs work on VM/ESA 1.2.2. For other releases, you may have to find out the correct counters in VMDBK and RDEVBLKS.

QMDCDSK EXEC

```
/* QMDCDSK EXEC */
$F1 = 'PF01'; $F2 = 'PF02'; $F3 = 'PF03'; $F4 = 'PF04'; $F5 = 'PF05'
$F6 = 'PF06'; $F7 = 'PF07'; $F8 = 'PF08'; $F9 = 'PF09'; $7A = 'PF10'
$7B = 'PF11'; $7C = 'PF12'; $C1 = 'PF13'; $C2 = 'PF14'; $C3 = 'PF15'
$C4 = 'PF16'; $C5 = 'PF17'; $C6 = 'PF18'; $C7 = 'PF19'; $C8 = 'PF20'
$C9 = 'PF21'; $4A = 'PF22'; $4B = 'PF23'; $4C = 'PF24'
$01 = 'PA1'; $6E = 'PA2'; $7D = 'ENTER'; $6D = 'CLEAR'

RDEVSER  = '0020'  /* OFFSET OF DISK LABEL                          */
RDEVCTSI = '00E0'  /* OFFSET OF COUNT OF I/O SOLICITED
                      INTERRUPTS IN RDEVBLOCKS                      */
RDEVMCIA = '0104'  /* OFFSET OF COUNT OF SSCH AVOIDED DUE TO
                      MINI-DISK CACHE READ HITS IN RDEVBLOCKS       */
/* THESE OFFSETS DEPEND ON VM/ESA VERSION AND LEVEL                 */
/* YOU MUST CHECK IN "CP DATA AREAS AND CONTROL BLOCKS - VOLUME 2" */
/* (LY24-5253) FOR THE CORRECT OFFSETS OF THE COUNTERS              */

HEADING_LINE_1 = ,
'2842F3'X||' DISK ',
'2842F1'X||'  DISK   ',
'2842F2'X||'    SIO EXECUTED   ',
'2842F4'X||'   SIO SAVED      ',
'2842F5'X||' SAVED/EXEC '
HEADING_LINE_2 = ,
'2842F3'X||' ADDR ',
'2842F1'X||'  LABEL  ',
```

```
'2842F2'X||'    TOTAL     DELTA',
'2842F4'X||'   TOTAL     DELTA',
'2842F5'X||'   RATIO    '
RESET_ALL:
HELP_WINDOW = ''
DISK_LINE. = ''
NSAVIO. = Ø
OSAVIO. = Ø
NCNTIO. = Ø
OCNTIO. = Ø
SORTIT = ' SORT 53.9 D '
DROP_DSK = Ø
STAM = TIME('R')
'VMFCLEAR'

LETSGO:
/* GET DISK INFO */
'PIPE CP Q DASD ALL',
'| SPLIT AT /,/',
'| NLOCATE /OFFLINE/',
'| NLOCATE /FREE/',
'| NLOCATE /ATTACHE/',
'| SPECS W2 1.5 ',
'| STEM DISK.'
DO I = 1 TO DISK.Ø
'PIPE CP LOCATE' DISK.I,
 '| DROP FIRST ',
 '| SPECS W1 STRIP 1',
 '| VAR RDEV_LOC'
LABEL_LOC = 'H'||D2X(X2D(RDEV_LOC)+X2D(RDEVSER))
IOCNT_LOC = 'H'||D2X(X2D(RDEV_LOC)+X2D(RDEVCTSI))
IOSAV_LOC = 'H'||D2X(X2D(RDEV_LOC)+X2D(RDEVMCIA))
'PIPE CP D 'LABEL_LOC||'.1Ø | SPECS W2 1 W3 9 | SPECS 1.12 X2C 1.8 | VAR
LABEL'
'PIPE CP D 'IOCNT_LOC' | SPECS WORD2 STRIP X2D 1.11 RIGHT | VAR IOCNT'
'PIPE CP D 'IOSAV_LOC' | SPECS WORD2 STRIP X2D 1.11 RIGHT | VAR IOSAV'
AHUZ = FORMAT((IOSAV/IOCNT)*1ØØ,6,2)||'%'
NSAVIO.I = STRIP(IOSAV)
NCNTIO.I = STRIP(IOCNT)
DELTB = NCNTIO.I-OCNTIO.I;IF DELTB = NCNTIO.I THEN DELTB = Ø
DELTA = NSAVIO.I-OSAVIO.I;IF DELTA = NSAVIO.I THEN DELTA = Ø
DISK_LIN.I = ,
'2842F3'X||DISK.I,
'2842F1'X||'  'LABEL,
'2842F2'X||IOCNT ,
'2842F2'X||RIGHT(DELTB,6),
'2842F4'X||IOSAV  ,
'2842F4'X||RIGHT(DELTA,6),
'2842F5'X||AHUZ        X2D(STRIP(DISK.I))
OSAVIO.I = STRIP(IOSAV)
```

```
OCNTIO.I = STRIP(IOCNT)
END
DISK_LIN.Ø = DISK.Ø
SORT_DROP_DISK_LINE:
'PIPE STEM DISK_LIN. ',
'| 'SORTIT,
'| DROP 'DROP_DSK ,
'| SPECS 1-87 1',
'| TAKE 2Ø',
'| STEM DISK_LINE.'

QMDC :
SCREEN = '8ØØ3'X||,
'11ØØ182903C02041F242F6'X||' MDC DASD PERFORMANCE DISPLAY
'||'11ØØ381DFØ'X||,
'11ØØ432903C020410042F2'X||DATE(E)||'11ØØ4C1DFØ'X||,
'11ØØ4F2903C02041F242F7'X||'INTERVAL: '||,
'11ØØ5A2903C02041F142F7'X||FORMAT(STAM,5,3)||'
SECONDS'||'11ØØ6F1DFØ'X||,
'11ØØ932903C020410042F2'X||TIME()||'11ØØ9C1DFØ'X||,
'11ØØ9F2903C02041F242ØØ'X||HEADING_LINE_1||'11ØØE41DFØ'X||,
'11ØØEF2903C02041F242ØØ'X||HEADING_LINE_2||'1101341DFØ'X||,
'1101402903C020410042F3'X||DISK_LINE.1 ||'1101831DFØ'X||,
'1101902903C020410042F3'X||DISK_LINE.2 ||'1101D31DFØ'X||,
'1101E02903C020410042F3'X||DISK_LINE.3 ||'1102231DFØ'X||,
'1102302903C020410042F3'X||DISK_LINE.4 ||'1102731DFØ'X||,
'1102802903C020410042F3'X||DISK_LINE.5 ||'1102C31DFØ'X||,
'1102D02903C020410042F3'X||DISK_LINE.6 ||'1103131DFØ'X||,
'1103202903C020410042F3'X||DISK_LINE.7 ||'1103631DFØ'X||,
'1103702903C020410042F3'X||DISK_LINE.8 ||'1103B31DFØ'X||,
'1103C02903C020410042F3'X||DISK_LINE.9 ||'1104031DFØ'X||,
'1104102903C020410042F3'X||DISK_LINE.10||'1104531DFØ'X||,
'1104602903C020410042F3'X||DISK_LINE.11||'1104A31DFØ'X||,
'1104B02903C020410042F3'X||DISK_LINE.12||'1104F31DFØ'X||,
'1105002903C020410042F3'X||DISK_LINE.13||'1105431DFØ'X||,
'1105502903C020410042F3'X||DISK_LINE.14||'1105931DFØ'X||,
'1105A02903C020410042F3'X||DISK_LINE.15||'1105E31DFØ'X||,
'1105F02903C020410042F3'X||DISK_LINE.16||'1106331DFØ'X||,
'1106402903C020410042F3'X||DISK_LINE.17||'1106831DFØ'X||,
'1106902903C020410042F3'X||DISK_LINE.18||'1106D31DFØ'X||,
'1106E02903C020410042F3'X||DISK_LINE.19||'1107231DFØ'X||,
'1107302903C020410042F3'X||DISK_LINE.2Ø||'1107731DFØ'X||,
HELP_WINDOW||,
'1107731DF02842F62841F2'X||'HELP = PFØ1'||'11077F1DFØ'X||,
'11ØØØØ13'X
'PIPE VAR SCREEN | FULLSCREEN ',
'| SPLIT AT ANYOF /'"11"X'/',
'| TAKE FIRST',
'| SPECS /$/ 1 1-1 C2X NEXT',
'| VAR AIDKEY'
```

```
HELP_WINDOW = ''
SELECT
  WHEN VALUE(AIDKEY) = 'PF12' | VALUE(AIDKEY) = 'PF24' THEN EXIT
  WHEN VALUE(AIDKEY) = 'PAØ2' | VALUE(AIDKEY) = 'CLEAR' THEN DO
   STAM = TIME('R')
   SIGNAL RESET_ALL
    END /* END RESET */

  WHEN VALUE(AIDKEY) = 'ENTER' THEN DO
   DROP_DSK = Ø;DROP_USR = Ø
   STAM = TIME('R')
   SIGNAL LETSGO
    END /* END ENTER */

  WHEN VALUE(AIDKEY) = 'PFØ2' THEN DO /* SORT BY DISK VIRTUAL ADDRESS */

   SORTIT = 'SORT 88.4 A'
   DROP_DSK = Ø
   SIGNAL SORT_DROP_DISK_LINE
    END /* END PFØ2 */
  WHEN VALUE(AIDKEY) = 'PF14' THEN DO /* SORT BY DISK VIRTUAL ADDRESS */

   SORTIT = 'SORT 88.4 D'
   DROP_DSK = Ø
   SIGNAL SORT_DROP_DISK_LINE
    END /* END PF14 */

  WHEN VALUE(AIDKEY) = 'PFØ3' THEN DO /* SORT BY DISK VOLUME ID       */

   SORTIT = 'SORT 15.6 A'
   DROP_DSK = Ø
   SIGNAL SORT_DROP_DISK_LINE
    END /* END PFØ3 */
  WHEN VALUE(AIDKEY) = 'PF15' THEN DO /* SORT BY DISK VOLUME ID       */

   SORTIT = 'SORT 15.6 D'
   DROP_DSK = Ø
   SIGNAL SORT_DROP_DISK_LINE
    END /* END PF15 */

  WHEN VALUE(AIDKEY) = 'PFØ4' THEN DO /* SORT BY EXECUTED SIOS        */

   SORTIT = 'SORT 29.9 D'
   DROP_DSK = Ø
   SIGNAL SORT_DROP_DISK_LINE
    END /* END PFØ4 */
  WHEN VALUE(AIDKEY) = 'PF16' THEN DO /* SORT BY EXECUTED SIOS        */

   SORTIT = 'SORT 29.9 A'
   DROP_DSK = Ø
```

```
 SIGNAL SORT_DROP_DISK_LINE
  END /* END PF16 */

WHEN VALUE(AIDKEY) = 'PFØ5' THEN DO /* SORT BY MDC-SAVED SIOS       */

 SORTIT = 'SORT 54.9 D'
 DROP_DSK = Ø
 SIGNAL SORT_DROP_DISK_LINE
  END /* END PFØ5 */
WHEN VALUE(AIDKEY) = 'PF17' THEN DO /* SORT BY MDC-SAVED SIOS       */

 SORTIT = 'SORT 54.9 A'
 DROP_DSK = Ø
 SIGNAL SORT_DROP_DISK_LINE
  END /* END PF17 */

WHEN VALUE(AIDKEY) = 'PFØ6' THEN DO /* SORT BY SAVED/EXECUTED RATIO */

 SORTIT = 'SORT 79.7 D'
 DROP_DSK = Ø
 SIGNAL SORT_DROP_DISK_LINE
  END /* END PFØ6 */
WHEN VALUE(AIDKEY) = 'PF18' THEN DO /* SORT BY SAVED/EXECUTED RATIO */

 SORTIT = 'SORT 79.7 A'
 DROP_DSK = Ø
 SIGNAL SORT_DROP_DISK_LINE
  END /* END PF18 */

WHEN VALUE(AIDKEY) = 'PFØ7' | VALUE(AIDKEY) = 'PF19' THEN DO
                                  /* SCROLL BACKWARD IN DISKS LIST */

 DROP_DSK = DROP_DSK-19
 IF DROP_DSK<Ø THEN DROP_DSK = Ø
 DISK_LINE. = COPIES(' ',4Ø)
 SIGNAL SORT_DROP_DISK_LINE
  END /* END PFØ7-PF19 */

WHEN VALUE(AIDKEY) = 'PFØ8' | VALUE(AIDKEY) = 'PFØ8' THEN DO
                                  /* SCROLL FORWARD  IN DISKS LIST */

 IF DROP_DSK+19<DISK.Ø THEN DROP_DSK = DROP_DSK+19
                            ELSE DROP_DSK = DISK.Ø-19
 DISK_LINE. = COPIES(' ',4Ø)
 SIGNAL SORT_DROP_DISK_LINE
  END /* END PFØ8-PF2Ø */

WHEN VALUE(AIDKEY) = 'PFØ1' | VALUE(AIDKEY) = 'PF13' THEN DO
                                   /* GET HELP WINDOW              */
```

```
HELP_WINDOW = '1102412903C02041F242F7'X||CENTER('HELP
WINDOW',33)||'1102631DF0'X||,
'1102912903C02041F242F7'X||' '||'1102931DF0'X||,
'1102932903C020410042F5'X||'PF01/PF13 : HELP            '||,
'1102B11DF0'X||'1102B12903C02041F242F7'X||' '||'1102B31DF0'X||,
'1102E12903C02041F242F7'X||' '||'1102E31DF0'X||,
'1102E32903C020410042F5'X||'PF02/PF14 : SORT BY ADDRESS  '||,
'1103011DF0'X||'1103012903C02041F242F7'X||' '||'1103031DF0'X||,
'1103312903C02041F242F7'X||' '||'1103331DF0'X||,
'1103332903C020410042F5'X||'PF03/PF15 : SORT BY LABEL    '||,
'1103511DF0'X||'1103512903C02041F242F7'X||' '||'1103531DF0'X||,
'1103812903C02041F242F7'X||' '||'1103831DF0'X||,
'1103832903C020410042F5'X||'PF04/PF16 : SORT BY EXEC SIO '||,
'1103A11DF0'X||'1103A12903C02041F242F7'X||' '||'1103A31DF0'X||,
'1103D12903C02041F242F7'X||' '||'1103D31DF0'X||,
'1103D32903C020410042F5'X||'PF05/PF17 : SORT BY SAVED SIO'||,
'1103F11DF0'X||'1103F12903C02041F242F7'X||' '||'1103F31DF0'X||,
'1104212903C02041F242F7'X||' '||'1104231DF0'X||,
'1104232903C020410042F5'X||'PF06/PF18 : SORT BY RATIO    '||,
'1104411DF0'X||'1104412903C02041F242F7'X||' '||'1104431DF0'X||,
'1104712903C02041F242F7'X||' '||'1104731DF0'X||,
'1104732903C020410042F5'X||'PF07/PF19 : SCROLL BACKWARD  '||,
'1104911DF0'X||'1104912903C02041F242F7'X||' '||'1104931DF0'X||,
'1104C12903C02041F242F7'X||' '||'1104C31DF0'X||,
'1104C32903C020410042F5'X||'PF08/PF20 : SCROLL FORWARD   '||,
'1104E11DF0'X||'1104E12903C02041F242F7'X||' '||'1104E31DF0'X||,
'1105112903C02041F242F7'X||' '||'1105131DF0'X||,
'1105132903C020410042F5'X||'PF09/PF21 :                  '||,
'1105311DF0'X||'1105312903C02041F242F7'X||' '||'1105331DF0'X||,
'1105612903C02041F242F7'X||' '||'1105631DF0'X||,
'1105632903C020410042F5'X||'PF10/PF22 :                  '||,
'1105811DF0'X||'1105812903C02041F242F7'X||' '||'1105831DF0'X||,
'1105B12903C02041F242F7'X||' '||'1105B31DF0'X||,
'1105B32903C020410042F5'X||'PF11/PF23 :                  '||,
'1105D11DF0'X||'1105D12903C02041F242F7'X||' '||'1105D31DF0'X||,
'1106012903C02041F242F7'X||' '||'1106031DF0'X||,
'1106032903C020410042F5'X||'PF12/PF24 : EXIT             '||,
'1106211DF0'X||'1106212903C02041F242F7'X||' '||'1106231DF0'X||,
'1106512903C02041F242F7'X||' '||'1106531DF0'X||,
'1106532903C020410042F5'X||'CLEAR/PA2 : RESET            '||,
'1106711DF0'X||'1106712903C02041F242F7'X||' '||'1106731DF0'X||,
'1106A12903C02041F242F7'X||' '||'1106A31DF0'X||,
'1106A32903C020410042F5'X||'ENTER : COMPUTE DELTA        '||,
'1106C11DF0'X||'1106C12903C02041F242F7'X||' '||'1106C31DF0'X||,
'1106F12903C02041F242F7'X||CENTER('HELP WINDOW',33)||'1107131DF0'X
   SIGNAL QMDC
    END /* END PF01-PF13 */

  OTHERWISE SIGNAL QMDC
END  /* END SELECT AIDKEY */
```

```
RETURN
/*
*/
```

QMDC EXEC

```
/* QMDC EXEC */
$F1 = 'PFØ1'; $F2 = 'PFØ2'; $F3 = 'PFØ3'; $F4 = 'PFØ4'; $F5 = 'PFØ5'
$F6 = 'PFØ6'; $F7 = 'PFØ7'; $F8 = 'PFØ8'; $F9 = 'PFØ9'; $7A = 'PF1Ø'
$7B = 'PF11'; $7C = 'PF12'; $C1 = 'PF13'; $C2 = 'PF14'; $C3 = 'PF15'
$C4 = 'PF16'; $C5 = 'PF17'; $C6 = 'PF18'; $C7 = 'PF19'; $C8 = 'PF2Ø'
$C9 = 'PF21'; $4A = 'PF22'; $4B = 'PF23'; $4C = 'PF24'
$Ø1 = 'PA1'; $6E = 'PA2'; $7D = 'ENTER'; $6D = 'CLEAR'

VMDCCSIO = 'ØD4'   /* OFFSET OF COUNT OF COMPLETED VIRTUAL I/O STARTS
                      TO REAL DEVICES OR MINI-DISKS, IN VMDBK        */

VMDMDCIA = '624'   /* OFFSET OF COUNT OF SSCH AVOIDED DUE TO
                      MINI-DISK CACHE READ HITS, IN VMDBK            */
/* THESE OFFSETS DEPEND ON VM/ESA VERSION AND LEVEL                  */
/* YOU MUST CHECK IN "CP DATA AREAS AND CONTROL BLOCKS - VOLUME 2" */
/* (LY24-5253) FOR THE CORRECT OFFSETS OF THE COUNTERS               */

HEADING_LINE_1 = ,
'2842F1'X||'  USER      ',
'2842F2'X||'   SIO ISSUED    ',
'2842F4'X||'    SIO SAVED     ',
'2842F5'X||'SAVED/ISSUED'
HEADING_LINE_2 = ,
'2842F1'X||'  NAME      ',
'2842F2'X||'   TOTAL    DELTA',
'2842F4'X||'    TOTAL    DELTA',
'2842F5'X||'   RATIO    '
RESET_ALL:
HELP_WINDOW = ''
USER_LINE. = ''
NSAVIO. = Ø
OSAVIO. = Ø
NCNTIO. = Ø
OCNTIO. = Ø
SORTIT = ' SORT 46.8 D '
DROP_USR = Ø
STAM = TIME('R')
'VMFCLEAR'
LETSGO:

/* GET USER INFO */
'PIPE CP Q N | SPLIT AFTER /,/',
'| NLOCATE /VSM/| NLOCATE /LOGN/| NLOCATE /LOGV/| NLOCATE /LOGL/',
```

```
'| CHANGE /-/ / | CHANGE /,/ / | SPECS WORD1 STRIP 1 | STEM USER.'
DO I = 1 TO USER.Ø
'PIPE CP LOCATE' USER.I,
 '| SPECS 12.5 1.5',
 '| VAR VMDBK_LOC'

IOCNT_LOC = 'H'||VMDBK_LOC||VMDCCSIO
IOSAV_LOC = 'H'||VMDBK_LOC||VMDMDCIA

'PIPE CP D 'IOCNT_LOC' | SPECS WORD2 STRIP X2D 1.11 RIGHT | VAR IOCNT'
'PIPE CP D 'IOSAV_LOC' | SPECS WORD2 STRIP X2D 1.11 RIGHT | VAR IOSAV'
IF IOCNT = Ø THEN AHUZ = '    Ø.ØØ%'
          ELSE AHUZ = FORMAT((IOSAV/IOCNT)*1ØØ,5,2)||'%'
NSAVIO.I = STRIP(IOSAV)
NCNTIO.I = STRIP(IOCNT)
DELTA = NSAVIO.I-OSAVIO.I ; IF DELTA = NSAVIO.I THEN DELTA = Ø
DELTB = NCNTIO.I-OCNTIO.I ; IF DELTB = NCNTIO.I THEN DELTB = Ø
USER_LIN.I = ,
'2842F1'X||LEFT(USER.I,1Ø),
'2842F2'X||IOCNT ,
'2842F2'X||RIGHT(DELTB,6),
'2842F4'X||IOSAV  ,
'2842F4'X||RIGHT(DELTA,6),
'2842F5'X||AHUZ
OSAVIO.I = STRIP(IOSAV)
OCNTIO.I = STRIP(IOCNT)
END
USER_LIN.Ø = USER.Ø
SORT_DROP_USER_LINE:
'PIPE STEM USER_LIN. ',
'| 'SORTIT,
'| DROP 'DROP_USR ,
'| TAKE 2Ø',
'| STEM USER_LINE.'
QMDC :
SCREEN = '8ØØ3'X||,
'11ØØ172903C02041F242F6'X||' MDC USERS PERFORMANCE DISPLAY
'||'1100371DFØ'X||,
'1100432903C020410042F2'X||DATE(E)||'11004C1DFØ'X||,
'11004F2903C02041F242F7'X||'INTERVAL: '||,
'11005A2903C02041F142F7'X||FORMAT(STAM,5,3)||'
SECONDS'||'11006F1DFØ'X||,
'1100932903C020410042F2'X||TIME()||'11009C1DFØ'X||,
'11009F2903C02041F242ØØ'X||HEADING_LINE_1||'1100DE1DFØ'X||,
'1100EF2903C02041F242ØØ'X||HEADING_LINE_2||'11012E1DFØ'X||,
'1101402903C020410042F3'X||USER_LINE.1 ||'1101831DFØ'X||,
'1101902903C020410042F3'X||USER_LINE.2 ||'1101D31DFØ'X||,
'1101E02903C020410042F3'X||USER_LINE.3 ||'1102231DFØ'X||,
'1102302903C020410042F3'X||USER_LINE.4 ||'1102731DFØ'X||,
'1102802903C020410042F3'X||USER_LINE.5 ||'1102C31DFØ'X||,
```

```
'1102D02903C020410042F3'X||USER_LINE.6 ||'1103131DF0'X||,
'1103202903C020410042F3'X||USER_LINE.7 ||'1103631DF0'X||,
'1103702903C020410042F3'X||USER_LINE.8 ||'1103B31DF0'X||,
'1103C02903C020410042F3'X||USER_LINE.9 ||'1104031DF0'X||,
'1104102903C020410042F3'X||USER_LINE.10||'1104531DF0'X||,
'1104602903C020410042F3'X||USER_LINE.11||'1104A31DF0'X||,
'1104B02903C020410042F3'X||USER_LINE.12||'1104F31DF0'X||,
'1105002903C020410042F3'X||USER_LINE.13||'1105431DF0'X||,
'1105502903C020410042F3'X||USER_LINE.14||'1105931DF0'X||,
'1105A02903C020410042F3'X||USER_LINE.15||'1105E31DF0'X||,
'1105F02903C020410042F3'X||USER_LINE.16||'1106331DF0'X||,
'1106402903C020410042F3'X||USER_LINE.17||'1106831DF0'X||,
'1106902903C020410042F3'X||USER_LINE.18||'1106D31DF0'X||,
'1106E02903C020410042F3'X||USER_LINE.19||'1107231DF0'X||,
'1107302903C020410042F3'X||USER_LINE.20||'1107731DF0'X||,
HELP_WINDOW||,
'1107711DF02842F62841F2'X||' HELP = PF01 '||'11077F1DF0'X||,
'11000013'X
'PIPE VAR SCREEN | FULLSCREEN ',
'| SPLIT AT ANYOF /'"11"X'/',
'| TAKE FIRST',
'| SPECS /$/ 1 1-1 C2X NEXT',
'| VAR AIDKEY'
HELP_WINDOW = ''
SELECT
  WHEN VALUE(AIDKEY) = 'PF12' | VALUE(AIDKEY) = 'PF24' THEN EXIT
  WHEN VALUE(AIDKEY) = 'PA02' | VALUE(AIDKEY) = 'CLEAR' THEN DO
   STAM = TIME('R')
   SIGNAL RESET_ALL
    END /* END RESET */

  WHEN VALUE(AIDKEY) = 'ENTER' THEN DO
   DROP_USR = 0
   STAM = TIME('R')
   SIGNAL LETSGO
    END /* END ENTER */

  WHEN VALUE(AIDKEY) = 'PF03' THEN DO   /* SORT BY USER NAME          */

   SORTIT = 'SORT  4.8 A'
   DROP_USR = 0
   SIGNAL SORT_DROP_USER_LINE
    END /* END PF03 */
  WHEN VALUE(AIDKEY) = 'PF15' THEN DO   /* SORT BY USER NAME          */

   SORTIT = 'SORT  4.8 D'
   DROP_USR = 0
   SIGNAL SORT_DROP_USER_LINE
    END /* END PF15 */
```

```
WHEN VALUE(AIDKEY) = 'PFØ4' THEN DO   /* SORT BY TOTAL ISSUED SIOS  */

 SORTIT = 'SORT 21.8 D'
 DROP_USR = Ø
 SIGNAL SORT_DROP_USER_LINE
  END /* END PFØ4 */
WHEN VALUE(AIDKEY) = 'PF16' THEN DO   /* SORT BY TOTAL ISSUED SIOS  */

 SORTIT = 'SORT 21.8 A'
 DROP_USR = Ø
 SIGNAL SORT_DROP_USER_LINE
  END /* END PF16 */

WHEN VALUE(AIDKEY) = 'PFØ5' THEN DO   /* SORT BY MDC-SAVED SIOS     */

 SORTIT = 'SORT 46.8 D'
 DROP_USR = Ø
 SIGNAL SORT_DROP_USER_LINE
  END /* END PFØ5 */
WHEN VALUE(AIDKEY) = 'PF17' THEN DO   /* SORT BY MDC-SAVED SIOS     */

 SORTIT = 'SORT 46.8 A'
 DROP_USR = Ø
 SIGNAL SORT_DROP_USER_LINE
  END /* END PF17 */

WHEN VALUE(AIDKEY) = 'PFØ6' THEN DO   /* SORT BY SAVED/ISSUED RATIO */

 SORTIT = 'SORT 7Ø.6 D'
 DROP_USR = Ø
 SIGNAL SORT_DROP_USER_LINE
  END /* END PFØ6 */
WHEN VALUE(AIDKEY) = 'PF18' THEN DO   /* SORT BY SAVED/ISSUED RATIO */

 SORTIT = 'SORT 7Ø.6 A'
 DROP_USR = Ø
 SIGNAL SORT_DROP_USER_LINE
  END /* END PF18 */

WHEN VALUE(AIDKEY) = 'PFØ7' | VALUE(AIDKEY) = 'PF19' THEN DO
                                  /* SCROLL BACKWARD IN USERS LIST */

 DROP_USR = DROP_USR-19
 IF DROP_USR<Ø THEN DROP_USR = Ø
 USER_LINE. = COPIES(' ',4Ø)
 SIGNAL SORT_DROP_USER_LINE
  END /* END PFØ7-PF19 */

WHEN VALUE(AIDKEY) = 'PFØ8' | VALUE(AIDKEY) = 'PFØ8' THEN DO
                                  /* SCROLL FORWARD  IN USERS LIST */
```

```
   IF DROP_USR+19<USER.Ø THEN DROP_USR = DROP_USR+19
                             ELSE DROP_USR = USER.Ø-19
   USER_LINE. = COPIES(' ',4Ø)
   SIGNAL SORT_DROP_USER_LINE
    END /* END PFØ8-PF2Ø */

  WHEN VALUE(AIDKEY) = 'PFØ1' | VALUE(AIDKEY) = 'PF13' THEN DO
                                   /* GET HELP WINDOW             */

HELP_WINDOW = '11Ø2412903CØ2Ø41F242F7'X||CENTER('HELP
WINDOW',33)||'11Ø2631DFØ'X||,
'11Ø2912903CØ2Ø41F242F7'X||' '||'11Ø2931DFØ'X||,
'11Ø2932903CØ2Ø410042F5'X||'PFØ1/PF13 : HELP            '||,
'11Ø2B11DFØ'X||'11Ø2B12903CØ2Ø41F242F7'X||' '||'11Ø2B31DFØ'X||,
'11Ø2E12903CØ2Ø41F242F7'X||' '||'11Ø2E31DFØ'X||,
'11Ø2E32903CØ2Ø410042F5'X||'PFØ2/PF14 :                 '||,
'11Ø3Ø11DFØ'X||'11Ø3Ø12903CØ2Ø41F242F7'X||' '||'11Ø3Ø31DFØ'X||,
'11Ø3312903CØ2Ø41F242F7'X||' '||'11Ø3331DFØ'X||,
'11Ø3332903CØ2Ø410042F5'X||'PFØ3/PF15 : SORT BY USERID  '||,
'11Ø3511DFØ'X||'11Ø3512903CØ2Ø41F242F7'X||' '||'11Ø3531DFØ'X||,
'11Ø3812903CØ2Ø41F242F7'X||' '||'11Ø3831DFØ'X||,
'11Ø3832903CØ2Ø410042F5'X||'PFØ4/PF16 : SORT BY IO ISSUED'||,
'11Ø3A11DFØ'X||'11Ø3A12903CØ2Ø41F242F7'X||' '||'11Ø3A31DFØ'X||,
'11Ø3D12903CØ2Ø41F242F7'X||' '||'11Ø3D31DFØ'X||,
'11Ø3D32903CØ2Ø410042F5'X||'PFØ5/PF17 : SORT BY IO SAVED '||,
'11Ø3F11DFØ'X||'11Ø3F12903CØ2Ø41F242F7'X||' '||'11Ø3F31DFØ'X||,
'11Ø4212903CØ2Ø41F242F7'X||' '||'11Ø4231DFØ'X||,
'11Ø4232903CØ2Ø410042F5'X||'PFØ6/PF18 : SORT BY RATIO   '||,
'11Ø4411DFØ'X||'11Ø4412903CØ2Ø41F242F7'X||' '||'11Ø4431DFØ'X||,
'11Ø4712903CØ2Ø41F242F7'X||' '||'11Ø4731DFØ'X||,
'11Ø4732903CØ2Ø410042F5'X||'PFØ7/PF19 : SCROLL BACKWARD '||,
'11Ø4911DFØ'X||'11Ø4912903CØ2Ø41F242F7'X||' '||'11Ø4931DFØ'X||,
'11Ø4C12903CØ2Ø41F242F7'X||' '||'11Ø4C31DFØ'X||,
'11Ø4C32903CØ2Ø410042F5'X||'PFØ8/PF2Ø : SCROLL FORWARD  '||,
'11Ø4E11DFØ'X||'11Ø4E12903CØ2Ø41F242F7'X||' '||'11Ø4E31DFØ'X||,
'11Ø5112903CØ2Ø41F242F7'X||' '||'11Ø5131DFØ'X||,
'11Ø5132903CØ2Ø410042F5'X||'PFØ9/PF21 :                 '||,
'11Ø5311DFØ'X||'11Ø5312903CØ2Ø41F242F7'X||' '||'11Ø5331DFØ'X||,
'11Ø5612903CØ2Ø41F242F7'X||' '||'11Ø5631DFØ'X||,
'11Ø5632903CØ2Ø410042F5'X||'PF1Ø/PF22 :                 '||,
'11Ø5811DFØ'X||'11Ø5812903CØ2Ø41F242F7'X||' '||'11Ø5831DFØ'X||,
'11Ø5B12903CØ2Ø41F242F7'X||' '||'11Ø5B31DFØ'X||,
'11Ø5B32903CØ2Ø410042F5'X||'PF11/PF23 :                 '||,
'11Ø5D11DFØ'X||'11Ø5D12903CØ2Ø41F242F7'X||' '||'11Ø5D31DFØ'X||,
'11Ø6Ø12903CØ2Ø41F242F7'X||' '||'11Ø6Ø31DFØ'X||,
'11Ø6Ø32903CØ2Ø410042F5'X||'PF12/PF24 : EXIT            '||,
'11Ø6211DFØ'X||'11Ø6212903CØ2Ø41F242F7'X||' '||'11Ø6231DFØ'X||,
'11Ø6512903CØ2Ø41F242F7'X||' '||'11Ø6531DFØ'X||,
'11Ø6532903CØ2Ø410042F5'X||'CLEAR/PA2 : RESET           '||,
```

```
'1106711DF0'X||'1106712903C02041F242F7'X||' '||'1106731DF0'X||,
'1106A12903C02041F242F7'X||' '||'1106A31DF0'X||,
'1106A32903C020410042F5'X||'ENTER : COMPUTE DELTA        '||,
'1106C11DF0'X||'1106C12903C02041F242F7'X||' '||'1106C31DF0'X||,
'1106F12903C02041F242F7'X||CENTER('HELP WINDOW',33)||'1107131DF0'X
   SIGNAL QMDC
    END /* END PF01-PF13 */

  OTHERWISE SIGNAL QMDC
END  /* END SELECT AIDKEY */
RETURN
/*
*/
```

Yaakov J Hazan
Technical Support Manager
Ynon Technologies & Computer Ltd

Multiplatform command scheduler – part 2

This month we continue the code that allows multiplatform command scheduling.

CLKQUEUE EXEC

```
/* AN MVI EXEC */
/* START CLKQUEUE IN CMS AT ORIGIN 60000  */
/* ALL DATE FORMATS MUST BE IN SORT FORMAT YY/MM/DD */
/* BUG: DATE ROLLOVERS DON'T RESET NEXT DAY START TIME RANGES. */
/* NEED TO USE THE REXXRDR LOGIC TO READ THE REXX QUEUE RULES. */
SYS=ADDRESS(); IF SYS = 'TSO' | SYS = 'MVS' THEN TSO = 1; ELSE TSO = 0
IF SYS = 'DOS' | SYS = 'KEDIT' THEN DOS = 1; ELSE DOS = 0
IF SYS = 'CMD' THEN OS2 = 1; ELSE OS2 = 0
IF SYS = 'CMS' | SYS = 'REXX' | SYS = 'XEDIT' THEN CMS = 1; ELSE CMS = 0
IF SYS = 'REXX' THEN ADDRESS CMS
PARSE UPPER ARG ARGSTRING
DEBUG = ''; X = (FIND(ARGSTRING,'*DEBUG'))
IF X ¬= 0 THEN DO; ARGSTRING = (DELWORD(ARGSTRING,X,1)); TRACE 'I'
    DEBUG = '*DEBUG'; END
IF (FIND(ARGSTRING,'?')) = 1 THEN SIGNAL DOC
IF TSO | CMS THEN XUSERID = USERID()
IF DOS THEN XUSERID = 'DOSUSER'
IF OS2 THEN XUSERID = 'OS2USER'
QUIET = 0; X = (FIND(ARGSTRING,'*QUIET'))
```

```
IF X ¬= Ø THEN DO; ARGSTRING = (DELWORD(ARGSTRING,X,1)); QUIET = 1; END
ONEPASS = Ø; X = (FIND(ARGSTRING,'*ONEPASS'))
IF X ¬= Ø THEN DO; ARGSTRING = (DELWORD(ARGSTRING,X,1));
                                                    ONEPASS = 1; END
CLKRULES = Ø; X = (FIND(ARGSTRING,'*CLKRULES'))
IF X ¬= Ø THEN DO; ARGSTRING = (DELWORD(ARGSTRING,X,1));
                                                    CLKRULES = 1;END
IF CLKRULES THEN CTLTYP = 'KBS'
IF ¬CLKRULES & (TSO | CMS) THEN CTLTYP = 'CLKQUEUE'
IF ¬CLKRULES & (OS2 | DOS) THEN CTLTYP = 'CTL'
IFT = ''; X = POS('*IFT(',ARGSTRING)    /* SEARCH FOR INPUT FILETYPE */
IF X ¬= Ø THEN DO
    Y = POS(')',ARGSTRING,X)              /* FIND END OF INPUT FIELD */
    IFT = SUBSTR(ARGSTRING,(X+5),(Y-X-5)) /* STORE IP FILENAME       */
    ARGSTRING = DELSTR(ARGSTRING,X,(Y-X+1)) /* DROP INPUT PARM FIELD */
    END
IF IFT ¬= '' THEN CTLTYP = IFT
LMTTIME = ''; X = POS('*TIMELMT(',TRANSLATE(ARGSTRING)); V = 9
IF X = Ø THEN DO
    X = POS('*LMT(',TRANSLATE(ARGSTRING)); V = 5
    END
IF X ¬= Ø THEN DO 1     /* PARAMETER SETTING IN THIS ROUTINE. */
    Y = POS(')',ARGSTRING,X); IF Y = Ø THEN LEAVE
    ZS = X + V; ZL = Y - X - V
    Z = STRIP(SUBSTR(ARGSTRING,ZS,ZL))
    IF DATATYPE(Z) <> 'NUM' THEN LEAVE
    LMTTIME = Z; TIMELMT = 'ØØ:ØØ:ØØ'
    ZL = Y - X + 1
    ARGSTRING = DELSTR(ARGSTRING,X,ZL)
    CALL 'TIMECALC' LMTTIME TIME() 'QUIET'
    IF RESULT <> Ø THEN LEAVE     /* STOP IMMEDIATELY AFTER LMT CHK. */
    PULL . . . . . TIMELMT .    /* CALCULATE WHEN TO STOP LOGIC    */
    SAY 'CLKQUEUE - TIME LIMIT SET FOR' TIMELMT',
                                          AND IT IS NOW' TIME()'.'
    END
IF ¬TSO THEN SIGNAL BEGIN
IPLIB = ''; X = POS('*IPLIB(',TRANSLATE(ARGSTRING)); V = 7
IF X = Ø THEN DO /* ADVANCED PARAMETER SETTING */
    X = POS('*LIB(',TRANSLATE(ARGSTRING)); V = 5
    END
U = X + V; W = U
IF X ¬= Ø THEN DO FOREVER    /* PARMREXX */
    Y = POS(')',ARGSTRING,U); IF Y = Ø THEN LEAVE
    Z = POS('(',ARGSTRING,W)  /* CHK FOR *VAL1(*SUB1(X) *SUB2(VAL) */
    IF Z ¬= Ø & Z < Y & LENGTH(ARGSTRING) > Y
       THEN DO; W = Z+1; U = Y+1; ITERATE; END
    ZS = X + V; ZL = Y - X - V
    IPLIB = STRIP(SUBSTR(LOWSTRING,ZS,ZL))
    ZL = Y - X + 1
    ARGSTRING = DELSTR(ARGSTRING,X,ZL)
```

```
    LEAVE
    END
/*********************************************************************/
BEGIN:
CTLQUE = ''
IF DATATYPE(WORD(ARGSTRING,1)) ¬= 'NUM'
   THEN PARSE VALUE ARGSTRING WITH CTLQUE SLPTIME SLPCYCLS Z
   ELSE PARSE VALUE ARGSTRING WITH SLPTIME SLPCYCLS Z
IF CTLQUE = ''
   THEN DO
    IF CMS | TSO THEN CTLQUE = 'CLKQUEUE'
    IF OS2 | DOS THEN CTLQUE = 'CLKQUEUE'
    END
IF TSO & IPLIB ¬= ''  /* WAS THE I/P DSN SPECIFIED? */
   THEN DO   /* YES, CHANGE TO TSO DSN FORMAT */
    CTLQUE = '$DSN'
    CTLTYP = IPLIB
    CTLMOD = 'QDD'
    END
IF SLPTIME ¬= '' & SLPCYCLS = '' THEN SLPCYCLS = 6Ø
IF SLPTIME = '' THEN SLPTIME = Ø
IF SLPCYCLS = '' THEN SLPCYCLS = Ø
IF (DATATYPE(SLPTIME,NUM)) ¬= 1 THEN SIGNAL ERRØ5Ø
IF (DATATYPE(SLPCYCLS,NUM)) ¬= 1 THEN SIGNAL ERRØ6Ø
SLPMINS = SLPTIME + Ø     /* DROP SUFFIX ZEROES */
IF SLPMINS > 99 THEN SIGNAL ERRØ7Ø
IF CMS THEN DO
    'REXXFST' CTLQUE CTLTYP '*'
    IF RC ¬= Ø THEN SIGNAL ERRØ1Ø
    PULL . . . . .  CTLMOD .
    'CMSQ QUERY DISK' LEFT(CTLMOD,1) '(STACK LIFO'
    PULL . . . Z .; PULL .; IF Z ¬= 'R/W' THEN SIGNAL ERRØ45
    END
RUNSEQ = Ø; RUNTOT = Ø; CNTCYCLS = Ø; PASSCNT = Ø
COLON = ''; RUNLIST = ''
BEGINQUE:
QUESEQ = Ø; WRTSEQ = Ø; PCESEQ = Ø
PASSCNT = PASSCNT + 1
READQUE:
IF LMTTIME ¬= '' THEN IF TIME() > TIMELMT THEN SIGNAL ERR26Ø
/*  SPECIAL INPUT FORMAT:
QUETAGB: 9Ø/11/12 Ø9:ØØ:ØØ 1.M1*1Ø:ØØ ØØ/ØØ/ØØ ØØ:ØØ:ØØ Ø,
        IF QUETAGA = Ø &,
           QUETAGB = Ø THEN DO;
            MSG OP THE LINE TEST AND DRIVE TEST RAN OK...;
            SENDFILE LINETEST HISTORY A OPERATOR;
            END
*/
IF PCESEQ > QUESEQ THEN QUESEQ = PCESEQ
QUESEQ = QUESEQ + 1
```

```
ACTDATE = DATE('O'); ACTTIME = TIME(); CLKREC = ''; COLON = '';
XCONT = Ø
DO PCE = Ø UNTIL (X = Ø & Y = Ø)
    IF CMS THEN 'REXXRDR' CTLQUE CTLTYP CTLMOD (QUESEQ+PCE)
    IF TSO THEN 'REXXRDR' CTLQUE CTLTYP 'A' (QUESEQ+PCE)
    IF OS2 | DOS THEN 'EXECIO 1 DISKR'
                                  CTLQUE'.'CTLTYP (QUESEQ+PCE) '( FINIS'
    IF RC ¬= Ø THEN DO
        IF QUEUED() ¬= Ø THEN PULL Z
        SIGNAL EOF
        END
    PULL CLKPCE
    CLKTXT = CLKPCE          /* SAVE THE ORIGINAL DATA FOR LATER  */
    CLKPCE = STRIP(CLKPCE)' '
    X = CLKPCE
    IF POS('/*',X) > Ø THEN DO UNTIL POS('/*',X) = Ø
        PARSE VAR X X '/*' Y '*/' Z
        IF Z ¬== '' THEN X =  STRIP(X||Z)' '
        END
    CLKPCE = X
    Z = POS(':',WORD(CLKPCE,1))
    IF PCE > Ø & Z > Ø THEN DO
        PCESEQ = QUESEQ + PCE - 1
        LEAVE PCE
        END
    X = LASTPOS(', ',CLKPCE)
    IF X ¬= Ø THEN DO
        IF SUBSTR(CLKPCE,X+1) = '' THEN,
                CLKREC = STRIP(CLKREC LEFT(CLKPCE,X-1))
           ELSE X = Ø
        END
    IF X ¬= Ø THEN XCONT = X
    Y = LASTPOS('; ',CLKPCE)
    IF Y ¬= Ø THEN DO
        IF SUBSTR(CLKPCE,Y+1) = '' THEN,
                CLKREC = STRIP(CLKREC LEFT(CLKPCE,Y))
           ELSE Y = Ø
        END
    IF Y ¬= Ø THEN XCONT = Ø
    IF PCE > Ø & SUBSTR(CLKPCE,3,1) = '/' & ( X = Ø & Y = Ø) THEN DO
        /* IF DATE FOUND MEANS NEXT CLKQUEUE COMMAND WAS REACHED. */
        PCESEQ = QUESEQ + PCE - 1
        LEAVE PCE
        END
    Z = PCE + 1              /* SET BECAUSE PCE COUNT STARTS AT Ø*/
    TXT.QUESEQ.Ø = Z         /* SAVE PIECE COUNT FOR UNLOADING TXT*/
    TXT.QUESEQ.Z = CLKTXT    /* SAVE THE ORIGINAL DATA IN CONTEXT */
    IF PCE > Ø & ( X = Ø & Y = Ø) & XCONT = Ø THEN DO
        /* CONTINUATIONS W/O ';' ARE ALLOWED THIS WAY */
        IF RIGHT(CLKREC,1) ¬= ';' THEN CLKREC = CLKREC';'
```

```
        END
    IF X = Ø & Y = Ø THEN DO
        CLKREC = STRIP(CLKREC CLKPCE)
        PCESEQ = QUESEQ + PCE
        END
    IF PCE = Ø &,
       LEFT(CLKREC,1) ¬= '*' &,
       (X ¬= Ø | Y ¬= Ø) THEN DO
        IF Z = Ø THEN IF DATATYPE(LEFT(CLKREC,1)) = 'NUM',
               THEN IF WORDS(CLKREC) ¬= 6 THEN CALL ERR2ØØ
                   ELSE NOP
               ELSE IF WORDS(CLKREC) ¬= 7 THEN CALL ERR2ØØ
                   ELSE NOP
        IF Z ¬= Ø THEN IF DATATYPE(LEFT(CLKREC,1)) = 'NUM',
               THEN CALL ERR2ØØ
               ELSE IF WORDS(CLKREC) ¬= 7 THEN CALL ERR2ØØ
        END
    END
QUETAG = '_'; TAGX = ''
IF DATATYPE(LEFT(CLKREC,1)) = 'NUM',
   THEN PARSE VAR CLKREC QUEDATE QUETIME QUEFACT,
            RUNDATE RUNTIME RUNCODE QUETEXT
   ELSE PARSE VAR CLKREC QUETAG QUEDATE QUETIME QUEFACT,
            RUNDATE RUNTIME RUNCODE QUETEXT
/* NEXT LINE LOOKS FOR PRIOR ERROR WITH CONTINUATIONS TO FLAG/PRT
THEM.*/
IF EC = 199 THEN SIGNAL SHWERREC
LSTDATE = RUNDATE; LSTTIME = RUNTIME
QUETAG = STRIP(QUETAG)
IF CLKREC = '' THEN SIGNAL READQUE            /* BLANK RECORD FOUND */
IF LEFT(QUETAG,1) = '*' THEN SIGNAL READQUE
COLON = ''
IF RIGHT(QUETAG,1) = ':' THEN DO
    COLON = ':'
    PARSE VAR QUETAG QUETAG ':' .
    END
IF QUETAG ¬= '_' THEN TAGX = QUETAG
IF QUETAG = 'EOF' THEN SIGNAL EOF
IF QUETAG = 'RULESEND' THEN SIGNAL EOF
IF QUETAG = 'INITQUES' THEN SIGNAL EOF
IF QUETAG = 'INITGOAL' THEN SIGNAL EOF
IF TAGX ¬= '' & DATATYPE(LEFT(QUETAG,1)) = 'NUM' THEN SIGNAL ERR19Ø
INTERPRET 'DROP' QUETAG
/* SAY '' VALUE(QUETAG) '=' RUNCODE (TO SEE VALS) */
IF DATATYPE(LEFT(QUETIME,1)) = 'CHAR' THEN DO
/* INSERT PLUS TIME SETTING LOGIC HERE.  EX. CMD1+M2Ø */
    NOP
    END
/* ALLOW USERS TO USE ANY VALID DATE OR TIME FORMAT. IE 12/24 OR 1Ø:ØØ*/
FSTDATE = QUEDATE; FSTTIME = QUETIME /*RESET AT WRT IF QUECYCLE IS SET*/
```

```
IF LENGTH(QUEDATE) ¬= 8 THEN DO
    FSTDATE = QUEDATE
    CALL 'REXXDATE' QUEDATE 'QUIET' DEBUG
    IF RESULT ¬= Ø THEN SIGNAL ERR21Ø   /*INVALID DATE FORMAT ENTERED.*/
    PULL STAR Z Z Z QUEDATE Z      /* MAKE FULLY QUALIFIED DATE FIELD */
    END
IF LENGTH(QUETIME) ¬= 8 THEN DO
    FSTTIME = QUETIME
    CALL 'TIMECALC' QUETIME 'QUIET' DEBUG
    IF RESULT ¬= Ø THEN SIGNAL ERR22Ø  /*INVALID TIME FORMAT ENTERED. */
    PULL STAR Z Z Z Z Z QUETIME Z   /*MAKE FULLY QUALIFIED TIME FIELD */
    END
XTIME = 'XTIME'
INTERPRET (QUETAG || '.XTIME =' "'" || QUETIME || "'")
INTERPRET VALUE(QUETAG) '= "'RUNCODE'"'
IF QUEDATE||'-'||QUETIME > ACTDATE||'-'||ACTTIME THEN SIGNAL READQUE
SETNEXT:
NXTDATE = QUEDATE; FIXDATE = ''; NXTTIME = QUETIME; CHKDATE = ''
MATHDATE = Ø; MATHTIME = Ø; QUESECS = Ø
QUECODE = ''; ENDDATE = ''; QUELOOP = ''; QUESTART = ''; QUESTOP = ''
PARSE VALUE QUEFACT WITH QUEDAYS '.' QUECYCLE
IF DATATYPE(QUEDAYS) ¬= 'NUM' THEN SIGNAL ERRØ9Ø
IF QUECYCLE = '' THEN SIGNAL SETDATE
PARSE VALUE QUECYCLE WITH QUECYCLE '*' QUELOOP
IF QUELOOP ¬= '' THEN PARSE VALUE QUELOOP WITH QUELOOP '.' ENDDATE
IF LENGTH(ENDDATE) = 8 THEN IF QUEDATE >= ENDDATE THEN DO
    QUEDROP = '*OK'
    IF PCESEQ ¬= QUESEQ THEN QUETEXT = ','
    CLKQUEUE = QUEDROP TAGX||COLON,
            QUEDATE QUETIME QUEFACT DATE('O') TIME() RUNCODE QUETEXT
    CALL WRITEREC
    SIGNAL READQUE
    END
QUECODE = LEFT(QUECYCLE,1)
QUESECS = SUBSTR(QUECYCLE,2,8)
IF QUECODE ¬= 'S' & QUECODE ¬= 'M' & QUECODE ¬= 'H' THEN SIGNAL ERRØ8Ø
IF DATATYPE(QUESECS) ¬= 'NUM' THEN SIGNAL ERRØ95
IF QUECODE = 'M' THEN QUESECS = QUESECS * 6Ø
IF QUECODE = 'H' THEN QUESECS = QUESECS * 36ØØ
IF QUESECS < 1 THEN SIGNAL ERR1ØØ
IF QUELOOP = '' THEN SIGNAL SETDATE
IF DATATYPE(QUELOOP) = 'NUM' THEN DO
    /* QUESTOP WON'T BE RECALCED EACH TIME FOR LOOP OPTION.*/
    PARSE VALUE RUNTIME WITH RUNTIME '-' QUESTART '-' QUESTOP
    END
QUESTOPH = ''; QUESTOPM = ''
IF DATATYPE(QUELOOP) = 'CHAR' THEN DO 1
    IF QUESTART ¬= '' THEN LEAVE
    PARSE VALUE QUELOOP WITH QUESTOPH ':' QUESTOPM ':' Z
    IF QUESTOPH = '' THEN QUESTOPH = 'ØØ'
```

```
    IF DATATYPE(QUESTOPH) ¬= 'NUM' THEN SIGNAL ERR160
    IF QUESTOPM = '' THEN QUESTOPM = 'ØØ'
    IF DATATYPE(QUESTOPM) ¬= 'NUM' THEN SIGNAL ERR166
    IF QUESTOPH > 61 | QUESTOPM > 6Ø THEN SIGNAL ERR166
    /* QUESTOP IS RECALCED EACH TIME FOR HH:MM OPTION.*/
    PARSE VALUE RUNTIME WITH RUNTIME '-' QUESTART '-' Z
    QUESTOP = (RIGHT(QUESTOPH,2))||':'||(RIGHT(QUESTOPM,2))
    QUESTOP = TRANSLATE(QUESTOP,'Ø',' ')
    END
IF QUESTART = '' & QUESTOPH = '' THEN DO
    QUESTART = LEFT(QUETIME,5)
    QUEWK = QUESECS * QUELOOP
    CALL 'TIMECALC' QUEWK QUESTART 'QUIET'
    IF RESULT ¬= Ø THEN SIGNAL ERR17Ø
    PULL Z STOPSECS Z Z Z Z QUESTOP Z
    QUESTOP = LEFT(QUESTOP,5)
    IF QUESTART > QUESTOP THEN SIGNAL ERR18Ø
    END
IF QUESTART = '' & QUESTOPH ¬= '' THEN DO
    QUESTART = LEFT(QUETIME,5)
    QUESTOP = (RIGHT(QUESTOPH,2))||':'||(RIGHT(QUESTOPM,2))
    QUESTOP = TRANSLATE(QUESTOP,'Ø',' ')
    IF QUESTART > QUESTOP THEN SIGNAL ERR188
    END
SETDATE:
IF NXTDATE = DATE('O') THEN CHKDATE = DATE('O')
CALL 'REXXDATE' NXTDATE 'QUIET'
IF RESULT ¬= Ø THEN SIGNAL ERRØ2Ø
PULL STAR Z MATHDATE Z
NEXTMATH = MATHDATE + QUEDAYS
CALL 'REXXDATE' NEXTMATH 'QUIET' DEBUG
IF RESULT ¬= Ø THEN SIGNAL ERRØ3Ø
PULL STAR Z Z Z SORTDATE Z
NXTDATE = (SUBSTR(SORTDATE,1,2))||'/'||(SUBSTR(SORTDATE,3,2))||,
       '/'||(SUBSTR(SORTDATE,5,2))
/* FOLLOWING NEW ROUTINE CALCS RATHER THEN LOOPS TO MAKE NXTDATE */
IF NXTDATE < DATE('O') & QUEDAYS ¬= Ø
   THEN DO
    CALL 'REXXDATE' NXTDATE 'QUIET' DEBUG
    IF RESULT ¬= Ø THEN SIGNAL ERRØ3Ø
    PULL STAR Z DAYSNXT Z
    CALL 'REXXDATE' DATE('O') 'QUIET' DEBUG
    IF RESULT ¬= Ø THEN SIGNAL ERRØ3Ø
    PULL STAR Z DAYSQUE Z
    DAYSDIF = DAYSQUE - DAYSNXT
    DAYSFAC = DAYSDIF % QUEDAYS
    DAYSADD = DAYSFAC * QUEDAYS
    DAYSNXT = DAYSNXT + DAYSADD
    /* NOW CHG MATH DATE BACK TO SORT DATE. */
    CALL 'REXXDATE' DAYSNXT 'QUIET' DEBUG
```

```
    IF RESULT ¬= Ø THEN SIGNAL ERRØ3Ø
    PULL STAR FULLDATE .
    NXTDATE = TRANSLATE('78612345',FULLDATE,'12345678')
    END
IF NXTDATE <= DATE('O') & QUEDAYS ¬= Ø THEN SIGNAL SETDATE
IF NXTDATE > DATE('O') & QUEDAYS = Ø THEN SIGNAL WRITEQUE
SETTIME:
NXTDYS = Ø
/* DO NOT RUN CMD UNLESS QUEUE DATE MATCHES CYCLE. */
IF CHKDATE = '' THEN DO
    QUESTART = ''; QUESTOP = ''; QUELOOP = ''
    /* CANCEL ANY START/STOP NOT RUN DAY */
    SIGNAL WRITEQUE
    END
IF QUECYCLE = '' THEN SIGNAL CHKQUEST
CALL 'TIMECALC' QUESECS NXTTIME 'QUIET' DEBUG
IF RESULT ¬= Ø THEN SIGNAL ERR11Ø
TIMEPULL:
PULL Z NXTSECS NXTDYS NXTHRS NXTMNS NXTSCS NXTTIME Z
FIXDATE = ''
IF NXTDYS > Ø THEN DO
    NEXTMATH = NEXTMATH + (NXTDYS - QUEDAYS)
    CALL 'REXXDATE' NEXTMATH 'QUIET'
    IF RESULT ¬= Ø THEN SIGNAL ERR12Ø
    PULL STAR Z Z Z SORTDATE Z
    FIXDATE = (SUBSTR(SORTDATE,1,2))||'/'||(SUBSTR(SORTDATE,3,2))||,
            '/'||(SUBSTR(SORTDATE,5,2))
    IF FIXDATE > NXTDATE THEN NXTDATE = FIXDATE
    SIGNAL CHKQUEST
    END
RUNTIME = TIME()
IF NXTTIME >= RUNTIME THEN SIGNAL CHKQUEST
CALL 'TIMECALC' 'Ø' RUNTIME 'QUIET'
IF RESULT ¬= Ø THEN SIGNAL ERR13Ø
PULL Z RUNSECS Z Z Z Z RUNTIME Z
DO N = NXTSECS BY QUESECS UNTIL N >= RUNSECS
    END
NXTSECS = N
CALL 'TIMECALC' NXTSECS 'Ø Ø Ø QUIET'
IF RESULT ¬= Ø THEN SIGNAL ERR14Ø
SIGNAL TIMEPULL
CHKQUEST:
RUNDATE = DATE('O'); RUNTIME = TIME()
RUNDTTM = RUNDATE||'-'||RUNTIME
/* DO NOT RUN CMD UNLESS QUEUE DATE MATCHES CYCLE. */
IF CHKDATE = '' THEN DO
    RUNDATE = LSTDATE; RUNTIME = LSTTIME
    SIGNAL WRITEQUE
    END
/* DO NOT RUN CMD IF I/P QUEDATE-QUETIME ARE LESS THAN RUNDATE-RUNTIME*/
```

```
IF LMTTIME ¬= '' THEN IF TIME() > TIMELMT THEN SIGNAL ERR26Ø
IF QUEDATE < RUNDATE THEN DO
    NXTDATE = CHKDATE; RUNDATE = LSTDATE; RUNTIME = LSTTIME
    IF QUESTART = '' THEN QUETIME = TIME()
                                    /* SET TO RUN AFTER PRESENT TIME*/
 /* FOLLOWING DO IS A SHORTCUT TO CALCULATING NXTTIME BY JUST LOOPING.*/
    IF QUECYCLE ¬= '' & QUELOOP = '' THEN DO
      /* IF NO START/STOP AND CYCLE SET THEN START AT BEGINNING OF DAY*/
        IF NXTTIME > QUETIME & QUESTART = '' THEN NXTTIME = 'ØØ:ØØ:ØØ'
        CALL 'TIMECALC' NXTTIME 'QUIET' DEBUG
        IF RESULT ¬= Ø THEN SIGNAL ERR23Ø  /* ERROR CALCING NXTTIME   */
        PULL STAR SECSNXT Z             /* SET NO. SECS IN NXTTIME     */
        CALL 'TIMECALC' QUETIME 'QUIET' DEBUG
        IF RESULT ¬= Ø THEN SIGNAL ERR24Ø  /* ERROR CALCING QUETIME   */
        PULL STAR SECSQUE Z             /* SET NO. SECS IN QUETIME     */
        SECSDIF = SECSQUE - SECSNXT    /* CALC DIFFERENCE NXT-QUE     */
        SECSFAC = SECSDIF % QUESECS    /* CALC # OF PERIODS TO QUETIME*/
        SECSADD = SECSFAC * QUESECS   /* CALC # OF SECS TO NXT NXTTIME*/
        CALL 'TIMECALC' SECSADD NXTTIME 'QUIET' DEBUG
        IF RESULT ¬= Ø THEN SIGNAL ERR25Ø
                                    /* ERROR CALCING NEW NXTTIME VAL */
        PULL . . . . . NXTTIME .    /* CALC NXTTIME 1 SHY OF QUETIME*/
        DO WHILE NXTTIME < QUETIME
            IF LMTTIME ¬= '' THEN IF TIME() > TIMELMT THEN EXIT 1Ø1
            CALL 'TIMECALC' QUESECS NXTTIME 'QUIET' DEBUG
            IF RESULT ¬= Ø THEN SIGNAL ERR11Ø
            PULL . . . . . NXTTIME .
            END
        END
    SIGNAL WRITEQUE
    END
IF QUEDAYS = Ø & QUECYCLE = '' THEN SIGNAL RUNQUEST
IF NXTDATE||'-'||NXTTIME > RUNDTTM THEN DO
    IF QUECYCLE ¬= '' & RUNTIME < QUETIME THEN SIGNAL WRITEQUE
    IF QUESTOP ¬= '' & (QUESTOP||':ØØ') < RUNTIME THEN SIGNAL WRITEQUE
    SIGNAL RUNQUEST
    END
IF QUECYCLE ¬= '' THEN SIGNAL SETTIME
SIGNAL SETDATE
```

Editor's note: this article will be continued next month.

Marc Vincent Irvin
Move Immediate Software (USA)

September 1994 – August 1997 index

Items below are references to articles that have appeared in *VM Update* since September 1994. References show the issue number followed by the page number(s). Individual copies of all issues from that date are available, as are a limited number of issues prior to September 1994.

Sterling Software has updated its VM:Webserver for OfficeVision with new functions for adding attachments to e-mail and scheduling meetings with a calendaring option, all accessible through Web browsers across multiple platforms.

Release 1.1 lets users attach files to notes sent, as well as view and save attachments received, regardless of the type of binary attachments, which include word processing documents and spreadsheets, audio, graphics, and video files.

Clearly aimed at sites that are facing big and potentially very expensive decisions regarding their in-house communications systems, the software provides a point and click interface for updating OfficeVision/VM.

Users can reply to notes in standard OfficeVision/VM form or in an Internet style, and they can tailor the signature that appears at the end of each note. The e-mail function is seen by Sterling as particularly important for 'schedule a meeting' users, since it allows for the scheduling, notifying, and reserving of conference rooms for multiple attendees at single and recurring meetings.

Also new are graphical front-ends providing a month at a glance and yearly month by month calendars that can be viewed on one screen and printed. Pages can also be customized to include company logos, site specific information, backgrounds, and text colours.
The company promises that no modifications to VM or OfficeVision are required, no conversion or migration of existing OfficeVision information is needed, and installation typically takes less than a day.

For further information contact:
Sterling Software, 1800 Alexander Bell Drive, Reston, VA 22091, USA.
Tel: (703) 264 8000.
Sterling Software International, 1 Longwalk Road, Stockley Park, Uxbridge, Middlesex, UB11 1DB, UK.
Tel: (0181) 867 8000.

* * *

Firesign Computer Company has announced Outbound SNA Server Professional, which enables high-speed unattended data transfer between VM, MVS, or VSE systems and Microsoft SNA servers.

Outbound SNA Server Professional runs as a service under a Windows NT server. New features include a new GUI, and it supports the simultaneous transfer of data to multiple LU connections.

The product also has two new tools. The administrator program provides real-time configuration access and allows administrators to set up and change the LUs used by the product. It also traces individual LUs for problem resolution and performance tuning. The monitor program provides a summary of Outbound's activity.

For further information contact:
Firesign Computer Company, 480 Green Street, San Francisco, CA 94133, USA.
Tel: (415) 398 7228.